Find the Hamming distance between two pairs of words

i) d (000, 011)
ii) d (10101, 11110)
iii) d (01010, 00010)
iv) d (1010, 0011)
v) d (1110,1011)

Here are the Hamming distances:

- **i) d(000, 011) = 2** because there are **two** differing bits.
- **ii) d(10101, 11110) = 3** because there are **three** differing bits.
- **iii) d(01010, 00010) = 1** because there is **one** differing bit.
- **iv) d(1010, 0011) = 3** because there are **three** differing bits.
- **v) d(1110, 1011) = 2** because there are **two** differing bits.

The Hamming distance is a measure of the difference between two words of equal length. It's calculated by counting the number of positions at which the corresponding bits are different.

Given the dataword 101001111 and the divisor 10111, show the generation of the CRC codeword at the sender site (using binary division).

Generating the CRC Codeword

Given the dataword **101001111** and the divisor **10111**, the CRC codeword can be generated at the sender site using the following steps:

1. **Augment the dataword:** Add (n-k) zeros to the right side of the dataword, where 'n' is the number of bits in the divisor (5 in this case) and 'k' is the number of bits in the dataword (9 in this case). This creates a dividend of 1010011100000.
2. **Perform modulo-2 binary division:** Divide the augmented dataword by the divisor (10111) using modulo-2 division. Modulo-2 division is

similar to regular binary division, but instead of subtracting, the XOR operation is used.

Here's how the division would look:

```
10100111100000  | 10111 (Divisor)
10111         | --------
-----        | 100010101 (Quotient)
 001111      |
  10111     |
  -----     |
   010011    |
   10111    |
   -----    |
    011000   |
     10111   |
     -----   |
      11110  |
      10111  |
      -----  |
       10111 |
       10111 |
       ----- |
        0000  |
```

3. **Obtain the remainder:** The remainder of this division is **0000**. This remainder represents the CRC check bits.
4. **Append the remainder to the dataword:** Append the CRC check bits (0000) to the original dataword (101001111) to form the codeword.

5. **Final codeword:** The final codeword is **10100111100000**. This codeword is then transmitted to the receiver.

Important Notes:

- The receiver performs the same modulo-2 division using the received codeword and the same divisor.
- If the remainder at the receiver is **0000**, it indicates no errors during transmission.
- If the remainder is not all zeros, it means that the received codeword has been corrupted during transmission.
- The specific divisor (10111 in this case) is chosen based on its error-detecting capabilities and is agreed upon by both sender and receiver beforehand.

A sender uses the stop and wait ARQ protocol for reliable transmission of frames. Frames are of size 1000 bytes and the transmission rate at the sender is 80 Kbps. Size of an acknowledgement is 100 bytes and the transmission rate at the receiver is 8 Kbps. The one way propagation delay is 100 msec.calculate sender throughput in bytes/sec assuming no frame is lost.

Here's how to calculate the sender throughput:

1. Calculate the transmission time for a frame (Tt)

- Frame size (L) = 1000 bytes = 8000 bits
- Transmission rate at the sender (R) = 80 Kbps = 80,000 bps
- Tt = L / R = 8000 bits / 80,000 bps = 0.1 seconds = 100 ms

2. Calculate the transmission time for an acknowledgment (Ta)

- Acknowledgment size (La) = 100 bytes = 800 bits
- Transmission rate at the receiver (Ra) = 8 Kbps = 8000 bps
- Ta = La / Ra = 800 bits / 8000 bps = 0.1 seconds = 100 ms

3. Calculate the total time for one cycle (Ttotal)

- One-way propagation delay (Tp) = 100 ms
- Ttotal = Tt + Tp + Ta + Tp = 100 ms + 100 ms + 100 ms + 100 ms = 400 ms

4. Calculate the sender throughput (S)

- Throughput is the amount of data successfully transmitted per unit of time.
- S = L / Ttotal = 1000 bytes / 0.4 seconds = **2500 bytes/sec**

Therefore, the sender throughput is 2500 bytes/sec assuming no frame is lost.

Explanation of Calculations

- **Transmission Time (Tt and Ta):** This is the time taken to send the actual bits of the frame or acknowledgment onto the transmission medium.
- **Propagation Delay (Tp):** This is the time a bit takes to travel from the sender to the receiver over the physical link.
- **Total Time (Ttotal):** This represents the total time for a complete cycle in Stop-and-Wait ARQ, which includes transmitting a frame, propagation to the receiver, receiving the acknowledgment, and propagation back to the sender.
- **Throughput (S):** This is the effective rate at which data is being transferred, considering the overheads of the protocol.

Key Points about Stop-and-Wait ARQ

- **Reliability:** Stop-and-Wait ARQ ensures reliable transmission by waiting for acknowledgment (ACK) before sending the next frame.
- **Simplicity:** It's relatively simple to implement.

- **Inefficiency:** It can be inefficient, especially for links with high bandwidth or long delays, as the sender remains idle while waiting for the ACK.

Match the following to one or more layers of the OSI model: a. Format and code conversion services b. Establishes, manages, and terminates sessions c. Reliable Process to Process delivery d. Log-in and log-out procedures e. Route selection

Here are the OSI layer matches for each of the services you listed:

- **a. Format and code conversion services: Presentation Layer (Layer 6)**. The presentation layer in the OSI model is responsible for translating data between the format used by the application layer and the format required for transmission over the network. This includes functions like data encryption, compression, and character encoding conversions. For example, it might handle converting ASCII text to Unicode or vice versa.
- **b. Establishes, manages, and terminates sessions: Session Layer (Layer 5)**. The session layer sets up, coordinates, and terminates conversations (sessions) between applications on different hosts. It manages things like dialog control (half-duplex or full-duplex) and checkpointing during data transfer, allowing for recovery from interruptions.
- **c. Reliable Process to Process delivery: Transport Layer (Layer 4)**. The transport layer is responsible for reliable end-to-end communication between processes running on different hosts. It can provide mechanisms for segmentation and reassembly of data, flow control, and error control to ensure that data arrives in order and without errors. TCP is a well-known example of a transport layer protocol that offers reliable process-to-process delivery.
- **d. Log-in and log-out procedures: Application Layer (Layer 7)**. Log-in and log-out procedures are specific to applications and how they handle user authentication. The application layer interacts directly

with users and provides services like email, file transfer, and remote access, which often involve log-in and log-out procedures.

- **e. Route selection: Network Layer (Layer 3)**. The network layer is responsible for deciding the path that data takes from the source host to the destination host. It uses logical addresses (like IP addresses) and routing tables to determine the optimal path for data packets.

Compare Stop and Wait ARQ, Go back N ARQ and Selective Repeat ARQ based on following points Efficiency, Window Size, Minimum number of sequence numbers required, Retransmissions required if a packet is lost, Bandwidth Requirement, CPU usage, Level of difficulty in Implementation, Acknowledgements, Type of Transmission

Comparing Stop and Wait ARQ, Go-back-N ARQ, and Selective Repeat ARQ

Here's a comparison of the three ARQ protocols based on the points you've listed:

Feature	Stop and Wait ARQ	Go-back-N ARQ	Selective Repeat ARQ
Efficiency	**Lowest**. The sender remains idle after transmitting a frame until it receives the ACK, wasting potential bandwidth. This inefficiency is more prominent on high-	**Moderate**. More efficient than Stop and Wait because multiple frames can be sent and acknowledged at a time, leading to better channel utilization. However, a single packet loss can cause the	**Highest**. Minimizes unnecessary retransmissions by selectively resending only the lost or corrupted packets. This leads to the most efficient use of the bandwidth,

	bandwidth or long-delay links (high bandwidth-delay product).	retransmission of multiple subsequent packets, even if they were correctly received.	especially on noisy links. However, it comes with increased complexity at the receiver.
Window Size	**Sender: 1, Receiver: 1**. The sender transmits only one frame at a time and waits for its acknowledgment before sending the next. The receiver only keeps track of the next expected frame.	**Sender: Up to (2m - 1), Receiver: 1**. The sender can transmit multiple frames within the window size, but the receiver still only maintains the next expected frame sequence number.	**Sender: Up to 2(m-1), Receiver: Up to 2(m-1)**. Both sender and receiver maintain windows of the same size. The sender's window limits outstanding frames, and the receiver's window buffers out-of-order frames.
Sequence Numbers	**Minimum: 2 (0 and 1)**. With a window size of 1, only two sequence numbers are needed for basic alternating-bit operation.	**Minimum: At least 2, ideally more depending on window size**. More sequence numbers are required as the window size increases (2m, where 'm' is the number of bits for sequence numbers).	**Minimum: At least 2, ideally more depending on window size**. Similar to Go-back-N, the number of sequence numbers increases with window size (2m, where 'm' is the number of bits for sequence numbers).

Retransmissions	**If a frame is lost, both the lost frame and any subsequent frames are retransmitted after a timeout.** The sender has no way to know which frame was lost and assumes all subsequent frames are also lost.	**Retransmits all unacknowledged frames in the window when the timer for the oldest unacknowledged frame expires.** Even if some frames were received correctly, they are retransmitted, leading to potential inefficiency.	**Only the lost or corrupted frame is retransmitted.** The receiver individually acknowledges each correctly received frame, allowing the sender to pinpoint and resend only the missing ones.
Bandwidth Req.	**Low**. Since only one frame is sent at a time, the bandwidth requirement is generally low, but inefficient use can make it seem higher.	**Moderate**. Allows for better bandwidth utilization than Stop and Wait, but retransmissions due to a single lost packet can increase the bandwidth consumption unnecessarily.	**High**. Optimizes bandwidth usage by minimizing retransmissions, allowing for more efficient data transfer, especially over noisy links where other methods would suffer.
CPU Usage	**Lowest**. Simple logic for sending, receiving, and timeouts.	**Moderate**. Requires management of sequence numbers, timers, and the send window. More complex logic for	**Highest**. More complex processing is needed at the receiver to handle out-of-order frames, maintain the receive

		handling acknowledgments and retransmissions than Stop and Wait.	window, and send individual acknowledgments.
Implementation Difficulty	**Easiest**. Straightforward logic and fewer variables to manage.	**Moderate**. More complex than Stop and Wait due to the sliding window and handling of multiple sequence numbers.	**Most difficult**. Requires more complex buffer management, sequence number handling, and acknowledgment mechanisms compared to the other two protocols.
Acknowledgments	**A single acknowledgment acknowledges only the specific frame indicated by the sequence number.**	**Cumulative acknowledgments**. A single ACK can acknowledge multiple frames up to a certain sequence number, improving efficiency.	**Individual acknowledgments for each correctly received frame.** This enables the selective retransmission of only the missing frames.
Type of Transmission	**Half-duplex**. The sender and receiver cannot transmit simultaneously.	**Half-duplex**. The sender generally waits for an ACK before transmitting more frames, even within its window size.	**Half-duplex**. Despite the ability to buffer out-of-order frames, the communication remains largely half-duplex, with the sender

primarily waiting for acknowledgments before transmitting further.

Additional Points:

- The choice of which ARQ protocol to use depends on factors like the characteristics of the communication channel (error rate, bandwidth, delay), complexity constraints, and performance requirements.
- Stop-and-Wait ARQ, while simple, is typically suitable for links with low bandwidth-delay products where its inefficiency is less significant.
- Go-back-N ARQ provides a balance between efficiency and implementation complexity and is often used in satellite communication links.
- Selective Repeat ARQ is preferred for noisy links with high bandwidth-delay products, where minimizing retransmissions is crucial, like in high-speed wireless networks.

Compare the TCP header and the UDP header. List the fields in the TCP header that are not part of the UDP header. Give the reason for each missing field.

Comparing TCP and UDP Headers

The key differences between the TCP and UDP headers reflect their different purposes and levels of complexity. Here's a breakdown:

Fields Present in TCP Header but Not in UDP Header:

- **Sequence Number (32 bits)**: TCP is a stream-oriented protocol, meaning it treats data as a continuous flow of bytes rather than individual messages. The sequence number field is crucial for:

  - **Reliable Data Transfer:** It allows the receiver to reassemble segments in the correct order, even if they arrive out of order due to network conditions.
  - **Flow Control:** By acknowledging received byte numbers, the receiver can control the rate at which the sender transmits data, preventing buffer overflow.
- **Acknowledgment Number (32 bits)**: This field contains the sequence number of the next byte the receiver expects, acknowledging receipt of all prior bytes. It's essential for TCP's reliable data transfer mechanism.
- **Receive Window Size (16 bits)**: This field specifies the number of bytes the receiver is willing to accept at that moment. It allows for dynamic flow control, where the receiver can adjust its buffer allocation based on current resources and network conditions.
- **Header Length (4 bits)**: TCP headers can have a variable length due to the optional "Options" field. This field indicates the header's length in 32-bit words, allowing the receiver to correctly parse the segment.
- **Urgent Data Pointer (16 bits)**: This field is used in conjunction with the URG flag to indicate data marked as "urgent." It points to the last byte of urgent data in the segment. While not commonly used, it demonstrates TCP's ability to prioritize certain data within a stream.
- **Flags (6 bits)**: TCP uses flags for connection management and data transfer control:
  - **SYN:** Synchronize sequence numbers, used during connection establishment.
  - **ACK:** Indicates that the acknowledgment number is valid.
  - **RST:** Reset the connection.
  - **FIN:** Indicates the sender has finished sending data.
  - **PSH:** Push data to the application layer immediately.
  - **URG:** Urgent data is present.
- **Options (Variable length)**: This optional field allows for:

    - **Maximum Segment Size (MSS) negotiation:** Determining the largest segment size acceptable for both sender and receiver.
    - **Window scaling:** For use in high-speed networks to extend the flow control window beyond its 16-bit limit.
    - **Timestamps:** For measuring round-trip times and improving reliability.

Reasons for Missing Fields in UDP Header:

The UDP header is intentionally minimalistic (only 8 bytes), reflecting its connectionless and unreliable nature. It lacks the fields present in the TCP header because it doesn't provide:

- **Reliable data transfer:** UDP doesn't guarantee in-order delivery or retransmission of lost packets.
- **Flow control:** UDP sends data without regard to the receiver's ability to process it.
- **Connection establishment and teardown:** UDP doesn't have the concept of a connection; it sends each datagram independently.

In summary: The TCP header includes additional fields to support its connection-oriented, reliable data transfer service, while the UDP header remains streamlined for applications where speed and simplicity are prioritized over reliability.

Elaborate with neat and clean diagram the different communication system with refrence to the following types - a. Hop-to-hop b. Host-to-host c. Process-to-process Also clearly mention the name of respective protocol layer, protocol data units used, and addressing scheme used in each type of communication.

Communication System Types: Hop-to-hop, Host-to-host, and Process-to-process

This response will elaborate on three types of communication systems with a focus on their respective protocol layers, protocol data units (PDUs), and addressing schemes.

1. Hop-to-Hop Communication

- **Definition:** This type of communication involves the direct exchange of data between two adjacent nodes (devices directly connected by a link) on a network. It is the responsibility of the **data link layer**.
- **Protocol Layer: Data Link Layer**
- **Protocol Data Unit (PDU): Frame**
- **Addressing Scheme: Physical Addressing (MAC Addresses)**: MAC addresses are unique identifiers assigned to network interface cards (NICs) and are used to identify a specific node on a local network segment.
- **Example:** When a computer wants to send data to another computer on the same Ethernet network, the data link layer encapsulates the data in a frame. This frame includes the source and destination MAC addresses. The frame is then transmitted over the shared Ethernet cable. Each computer on the network receives the frame, but only the computer with the matching destination MAC address processes it.
- **Diagram:** The sources do not include a diagram specifically for hop-to-hop communication. However, Figure 2.11 in the sources illustrates the concept of moving data across multiple hops, with each hop representing hop-to-hop delivery managed by the data link layer.

2. Host-to-Host Communication

- **Definition:** This level of communication focuses on delivering data from the source host (originating computer) to the destination host (receiving computer), potentially across multiple interconnected networks. The **network layer** is responsible for this type of communication.
- **Protocol Layer: Network Layer**

- **Protocol Data Unit (PDU): Datagram** (also referred to as a packet at the network layer)
- **Addressing Scheme: Logical Addressing (IP Addresses)**: IP addresses are used to identify a specific host on a network, whether it's a local network or the internet.
- **Example:** When you send an email, your email client uses the network layer to send the email message to your email server. The network layer encapsulates the message in a datagram (IP packet) that includes the source and destination IP addresses. The datagram is then routed through various routers across the internet to reach the destination server.
- **Diagram:** Figure 20.3c provides a representation of a router in a network. Routers are essential components in host-to-host communication as they utilize routing tables to determine the best path for datagrams to travel between hosts.

3. Process-to-Process Communication

- **Definition:** This is the highest level of communication and deals with the exchange of data between specific software applications (processes) running on different hosts. It is the responsibility of the **transport layer**.
- **Protocol Layer: Transport Layer**
- **Protocol Data Unit (PDU): Segment** (also referred to as a datagram in some contexts)
- **Addressing Scheme: Port Addressing:** Port numbers are used to identify specific processes or services running on a host. They allow multiple applications on the same host to send and receive data simultaneously.
- **Example:** When you visit a website, your web browser uses the HTTP protocol (an application-layer protocol) to communicate with the web server. HTTP relies on the transport layer protocol TCP to establish a connection and reliably exchange data. TCP uses port numbers to differentiate between the web browser process on your computer and the web server process on the remote server.

- **Diagram:** Figure 23.1 depicts the relationship between process-to-process, host-to-host, and node-to-node communication, highlighting the transport layer's role in delivering segments between processes.

Draw the basic diagram for data communications system, and discuss three network criterias necessary for an effective and efficient network design.

Data Communication System Diagram and Network Design Criteria

Basic Diagram of a Data Communication System:

The core components of a data communications system are as follows:

- **Message:** The data being transmitted, which can be text, numbers, images, audio, video, or any other form of information.
- **Sender:** The device that originates the message and transmits it, such as a computer, smartphone, or sensor.
- **Receiver:** The device that receives the message, which could be a computer, smartphone, server, or other types of devices.
- **Transmission Medium:** The physical path over which the message travels, including copper wire, fiber optic cable, wireless radio waves, or satellite links.
- **Protocol:** A set of rules or procedures that govern the format and transmission of data between devices. Protocols ensure that devices can communicate effectively and exchange data reliably.

Three Essential Network Criteria:

To design effective and efficient networks, you must consider many crucial criteria. Three of the most critical are:

- **Performance:** This refers to the network's ability to transfer data quickly and efficiently. Performance can be measured using metrics like:

    - **Transit time:** The amount of time it takes for a message to travel from the sender to the receiver.
    - **Response time:** The time between a request being sent and a response being received.
    - **Throughput:** The amount of data that can be transmitted over the network in a given time period.
    - **Delay:** The latency experienced by data traveling across the network.

It's important to strive for high throughput and low delays, although these two goals often conflict. Increasing the volume of data transmitted can boost throughput but might lead to congestion and increased delay. Network designers must carefully balance these factors to optimize performance for specific applications and user requirements.

- **Reliability:** This metric measures the network's ability to consistently and dependably transmit data without errors or failures. Key aspects of reliability include:
    - **Frequency of failure:** How often the network or its components experience outages.
    - **Recovery time:** How quickly the network can restore connectivity after a failure.
    - **Robustness:** The network's ability to function even in the face of component failures or challenging environmental conditions.
- **Security:** Network security encompasses a range of measures aimed at safeguarding data and resources from unauthorized access, use, disclosure, disruption, modification, or destruction. Key security principles include:
    - **Confidentiality:** Ensuring that data is accessible only to authorized users and entities.
    - **Integrity:** Protecting data from unauthorized alteration or modification, ensuring that it remains accurate and trustworthy.

- **Availability:** Making sure that network resources and services are available to legitimate users when needed.

Network designers must implement security mechanisms, protocols, and policies to mitigate risks and protect sensitive information from various threats.

A bit stream 10011101 is transmitted using the standard CRC method. The generator polynomial is x3+1. 1. What is the actual bit string transmitted? 2. Suppose the third bit from the left is inverted during transmission. How will receiver detect this error?

CRC Calculation and Error Detection

Here's how to determine the transmitted bit string and explain error detection using CRC:

1. Calculating the Transmitted Bit String

- **Step 1: Represent the generator polynomial in binary.** The polynomial $x^3 + 1$ translates to the binary pattern 1001 (1 for the x^3 term, 0 for x^2 and x, and 1 for the constant term).
- **Step 2: Append zeros to the data bit stream.** The generator polynomial has a degree of 3 (highest power of x), so append three zeros to the data bit stream: 10011101 becomes 10011101000.
- **Step 3: Perform modulo-2 division.** Divide the modified data bit stream (10011101000) by the generator polynomial (1001) using modulo-2 division (XOR operation for subtraction).
- 10011011   <-- Quotient (Discarded)
- ------------
- 1001 | 10011101000 <-- Modified data bit stream
- 1001
- ----

- 0111
- 1001
- ----
- 1100
- 1001
- ----
- 1011
- 1001
- ----
- 0100
- 0000
- ----
- 1000 <-- Remainder (CRC bits)
- **Step 4: Append the remainder (CRC bits) to the original data bit stream.** The remainder (1000) is appended to the original data (10011101), resulting in the transmitted bit string: 100111011000.

2. Error Detection at the Receiver

- The receiver receives the transmitted bit string (potentially with errors) and divides it by the same generator polynomial (1001) using modulo-2 division.
- **Scenario: Third bit from the left inverted.** If the third bit from the left is inverted during transmission, the received bit string would be 101111011000. Performing modulo-2 division:
- 10111100 <-- Quotient (Discarded)
- ------------
- 1001 | 101111011000 <-- Received bit string
- 1001
- ----

- 0101
- 1001
- ----
- 0101
- 1001
- ----
- 00100
- 1001
- ----
- 00110
- 00000
- ----
- 01100
- 1001
- ----
- 1110
- 1001
- ----
- 1110 <-- Remainder (Non-zero)
- **Result:** Since the remainder is non-zero (1110), the receiver detects an error in the transmitted data. The sources do not explicitly mention the detection of errors beyond stating that a non-zero remainder indicates an error.

Key Points About CRC:

- CRC is an effective method for detecting burst errors.
- The choice of generator polynomial is crucial for determining the types and lengths of errors that can be detected.

- The sources you provided don't detail how the receiver handles detected errors, but generally, the receiver would discard the corrupted frame and rely on higher-level protocols for retransmission.

Determine codeword using hamming code for the given data bits 10011010.

Determining the Hamming Code Codeword

Here's how to determine the Hamming code codeword for the given data bits, keeping in mind that the sources primarily focus on error detection and correction mechanisms:

1. Determine the Number of Parity Bits (r) and Codeword Length (n)

- The sources provide the relationship between the number of data bits (k) and the number of parity bits (r) for Hamming codes: n = 2m - 1 and k = n - m, where m is an integer greater than or equal to 3.
- You have k = 8 data bits (10011010). You need to find an 'm' that satisfies the equation k = n - m = 2m - 1 - m.
- By trial and error, you'll find that m = 4 satisfies this condition (24 - 1 - 4 = 8).
- Therefore, you need r = m = 4 parity bits, and the codeword length will be n = 2m - 1 = 15.

2. Determine Parity Bit Positions

- The sources don't provide a general method for placing parity bits within the codeword. However, they illustrate how parity bits are calculated based on specific data bit positions.

3. Calculate Parity Bits

- The sources illustrate parity bit calculation for a Hamming code C(7, 4) example. However, they don't specify a general algorithm or formula for calculating parity bits for any Hamming code.

4. Construct the Codeword

- Without specific rules from the sources on parity bit placement and calculation for any Hamming code, constructing the codeword for your given data bits (10011010) and chosen parameters (n=15, r=4) is not possible using only the information in the sources.

Important Note: The sources provided focus on the general concepts of error detection and correction using Hamming codes, providing specific examples for illustration. They lack the precise algorithmic details necessary to directly calculate the codeword for your given data bits.

You have two computers connected by ethernet hub , is this a LAN,MAN, or WAN? Justify the reason.

This network is **a LAN**. A LAN, or local area network, is a network that connects devices in a small geographical area like a single office, building, or campus. The defining characteristic of a LAN is its limited size.

- Connecting two computers with an ethernet hub at home means the network spans an extremely limited area.
- While WANs cover a significantly larger area like states, countries, or the whole world, a MAN is a network with a size between a LAN and a WAN, usually covering a town or city.
- Based on the relative sizes of LANs, MANs, and WANs, and given the limited area covered by two computers connected in a home, the network is a LAN.

What is Piggybacking in data link layer?

Piggybacking is a technique used in data link layer protocols to improve the efficiency of bidirectional communication. The sources elaborate on this concept in the context of data link control and flow control.

How Piggybacking Works:

- In typical data link layer protocols, frames often travel in both directions between communicating nodes.
- Without piggybacking, acknowledgements (ACKs) and negative acknowledgements (NAKs) for received frames are sent as separate control frames.
- Piggybacking optimizes this process by including ACK or NAK information within data frames that are already being transmitted.
- For instance, if Node A sends a data frame to Node B, Node B can include an ACK for Node A's frame in the data frame it sends back to Node A. This combined frame carries both data and control information, reducing the need for separate control frames and improving network efficiency.

Piggybacking in Bidirectional Protocols:

- While simpler unidirectional protocols may not employ piggybacking, it's a common feature in more realistic bidirectional protocols where data flows in both directions simultaneously.
- Implementing piggybacking in bidirectional scenarios is more complex as it requires coordinating the arrival of data frames and incorporating ACK/NAK information for frames received from both directions.

Example of Piggybacking:

- Imagine Node A sends frames 0 and 1 to Node B.
- Node B receives these frames and intends to send its own data frame back to Node A.
- Using piggybacking, Node B can include an ACK for both frames 0 and 1 within its data frame, indicating to Node A that it received the frames correctly and expects the next frame in the sequence (frame 2).

Key Points about Piggybacking:

- **Efficiency:** By combining data and control information, piggybacking reduces the overhead of separate control frames, optimizing bandwidth use and improving overall network efficiency.
- **Bidirectional Communication:** It is particularly beneficial in situations where data exchange occurs in both directions, making communication smoother.
- **Algorithm Agreement:** Successful implementation of piggybacking requires both communicating nodes to use the same algorithm for encoding and decoding the ACK/NAK information within data frames.

Compute the cyclic redundancy check (CRC) for this message M = 1010001101 using the divisor polynomial x5 + x4 + x2 + 1.

Computing the Cyclic Redundancy Check (CRC)

Here's how to calculate the CRC for the message M = 1010001101 using the divisor polynomial $x^5 + x^4 + x^2 + 1$, drawing on the information provided in the sources:

1. Represent the Polynomials in Binary

- **Message (M):** 1010001101
- **Divisor ($x^5 + x^4 + x^2 + 1$):** 110101

2. Append Zeros to the Message

- The divisor polynomial has a degree of 5, so append five zeros to the message: 1010001101 becomes **101000110100000**.

3. Perform Modulo-2 Division

- Divide the modified message by the divisor polynomial using modulo-2 division (XOR operation for subtraction).

```
   110000110 <-- Quotient (Discarded)

  ------------

110101 | 101000110100000 <-- Modified Message

     110101

     ------
```

```
1111001

110101

------

 1001110

 110101

 ------

  10011100

  110101

  ------

   10101100

   110101

   ------

    11001100

    110101

    ------

     110100  <-- Remainder (CRC bits)
```

4. The CRC

- The remainder obtained from the modulo-2 division, **110100**, represents the CRC bits for the given message and divisor polynomial.

Important Notes about CRC:

- **Error Detection:** The CRC is used for error detection in digital networks. The receiver performs the same CRC calculation upon receiving the message and compares the calculated CRC with the received CRC. If they match, the transmission is considered error-free (within the detection capabilities of the specific CRC polynomial).
- **Generator Polynomial Selection:** The choice of generator polynomial significantly impacts the types and lengths of errors the

CRC mechanism can detect. Standard polynomials, like those listed in Table 10.7 in the sources, are often chosen based on their error-detection properties.

Write Detection versus Correction with example for each difference

Error Detection vs. Error Correction

The sources distinguish between two primary approaches to handling errors in data transmission: **detection** and **correction**. While both rely on the principle of redundancy (adding extra bits to the data), their goals and methods differ.

Error Detection

- **Goal:** To determine if any errors occurred during transmission. The receiver doesn't need to know the specific errors or their locations, just whether the received data is corrupted.
- **Method:** The sender applies an error-detection code, like a parity check or cyclic redundancy check (CRC), to generate redundant bits based on the data. The receiver performs the same calculation on the received data and compares its result with the received redundant bits. A mismatch indicates an error.
- **Example:** In our previous conversation, we calculated the CRC for a message using a specific divisor polynomial. A receiver, using the same polynomial, would independently calculate the CRC upon receiving the message. If the calculated CRC doesn't match the received CRC, the receiver knows an error occurred.
- **Action:** Upon detecting an error, the receiver typically discards the corrupted data. The sources do not detail the error-handling process beyond detection, but generally, higher-level protocols (like those in the transport layer) would manage retransmission.

Error Correction

- **Goal:** To identify the specific bit errors in the received data and correct them.

- **Method:** The sender encodes the data using an error-correcting code, like a Hamming code, which adds more redundant bits than error-detecting codes. These extra bits enable the receiver to pinpoint and rectify a limited number of errors.
- **Example:** Suppose you use a Hamming code capable of correcting single-bit errors. If one bit in the received codeword is flipped, the receiver can, using the properties of the Hamming code, identify the exact position of the incorrect bit and flip it back to its original value, thereby correcting the error.
- **Action:** The receiver, upon successful error correction, reconstructs the original data without requiring retransmission.

Key Differences Summarized

Feature	Error Detection	Error Correction
Primary Goal	Identify if errors occurred	Correct identified errors
Complexity	Simpler; fewer redundant bits	More complex; more redundant bits
Overhead	Lower	Higher
Receiver Action	Discard corrupted data, rely on retransmission	Reconstruct original data

Important Considerations:

- **Error Types:** The effectiveness of both methods depends on the types and frequency of errors encountered. Burst errors, where multiple consecutive bits are corrupted, are more challenging to handle, especially for correction.
- **Trade-offs:** Error detection is simpler and incurs less overhead, but it relies on retransmission, which increases delay. Error correction eliminates retransmission overhead but adds complexity and increases data size due to the greater number of redundant bits.

For n devices in a network, what is the number of cable links required for a mesh,ring, bus, and star topology?

Here's a breakdown of the number of cable links required for different network topologies, based on information from the sources you provided:

Cable Links for Different Topologies

- **Mesh Topology:** In a mesh topology, every device has a dedicated point-to-point link to every other device in the network. To calculate the number of links for 'n' devices, consider that each device needs 'n-1' connections. However, since each link is bidirectional (duplex mode), we divide the initial calculation by 2. Therefore, the number of cable links required for a mesh topology with 'n' devices is **n(n-1)/2**.
- **Ring Topology:** A ring topology arranges devices in a closed loop, where each device connects to its two immediate neighbors. Since each connection serves two devices, the number of cable links in a ring topology is equal to the number of devices, or **n**.
- **Bus Topology:** In a bus topology, all devices share a single cable, often referred to as the backbone or segment. Only **one** cable is used, regardless of the number of devices connected.
- **Star Topology:** A star topology connects each device to a central hub using a dedicated point-to-point link. Similar to a ring topology, the number of cable links in a star topology directly corresponds to the number of devices, resulting in **n** links.

Summary Table:

Topology	Cable Links
Mesh	n(n-1)/2
Ring	n
Bus	1
Star	n

For each of the following four networks, discuss the consequences if a connection fails. a. Five devices arranged in a mesh topology b. Five devices arranged in a star topology (not counting the hub/Switch) c. Five devices arranged in a bus topology d. Five devices arranged in a ring topology

Here's a discussion of the consequences of connection failures in different network topologies:

Connection Failure Consequences in Different Topologies

- **a. Five devices arranged in a mesh topology:** In a mesh topology, with its inherent redundancy (every device connected to every other device), a single connection failure doesn't disrupt the entire network. The remaining devices can still communicate, although messages might have to be rerouted. For instance, if the link between Device 1 and Device 3 fails, Device 1 can still send data to Device 3 via an alternative path, such as Device 1 -> Device 2 -> Device 3. This robustness makes mesh networks suitable for applications where high availability is critical.
- **b. Five devices arranged in a star topology (not counting the hub/Switch):** The sources highlight a significant drawback of star topologies: the reliance on a central hub. If a connection between a device and the central hub fails, that specific device gets isolated from the network. For example, if the link between Device 2 and the hub breaks in a five-device star network, Device 2 can't send or receive data anymore, even though the other four devices remain connected.
- **c. Five devices arranged in a bus topology:** A bus topology, where all devices share a single cable, is highly susceptible to connection failures. A break or disruption anywhere along the shared cable can halt communication for the entire network. For instance, if the main cable in a five-device bus network is severed, none of the devices can communicate with each other until the cable is repaired.
- **d. Five devices arranged in a ring topology:** The impact of a connection failure in a ring topology depends on its configuration. In a simple ring, a single break in the connection can disrupt the entire network, as data flow is unidirectional. However, more sophisticated ring topologies, like the dual-ring setup used in FDDI and CDDI, provide redundancy to mitigate this vulnerability. A dual-ring employs a second ring operating in the reverse direction. If a link in the primary

ring fails, the dual-ring configuration automatically reconfigures, using the second ring to maintain connectivity.

Why the window sizes at sender and receiver sides are different in Go-Back-N ARQ and Selective Repeat ARQ protocols. Also, mention the window sizes of both the protocols.

Window Sizes in Go-Back-N and Selective Repeat ARQ

The sources explain that the window sizes at the sender and receiver sides are different in Go-Back-N ARQ and Selective Repeat ARQ protocols due to the distinct ways these protocols handle out-of-order packets and acknowledgements.

Go-Back-N ARQ

- **Sender Window Size:** The sender window size in Go-Back-N ARQ is **less than 2m**, where 'm' is the number of bits used for sequence numbering. The sources illustrate that this limitation on the sender window size ensures proper operation of the protocol, particularly in scenarios where acknowledgements are lost. If the window size were equal to or greater than 2m, the receiver might mistakenly accept a retransmitted frame as a new frame, leading to errors.
- **Receiver Window Size:** The receiver window size in Go-Back-N ARQ is always **1**. This means the receiver only accepts the next expected frame in sequence and discards any out-of-order frames it receives. The sender, upon timeout or receiving a NAK (negative acknowledgment), resends all unacknowledged frames starting from the one with the expired timer.

Selective Repeat ARQ

- **Sender Window Size:** The sender window size in Selective Repeat ARQ is **2m-1**. This smaller sender window size, compared to Go-Back-N, reflects the protocol's approach of selectively retransmitting only the frames suspected to be lost or corrupted.
- **Receiver Window Size:** The receiver window size in Selective Repeat ARQ is the same as the sender window size, which is **2m-1**. Unlike Go-Back-N, Selective Repeat allows the receiver to buffer out-of-

order frames within the window's range. The receiver only acknowledges frames that are received in order and delivered to the upper layer. This method avoids the redundant retransmissions characteristic of Go-Back-N. However, it requires a larger receive buffer to store out-of-order packets.

Summary Table:

Protocol	**Sender Window Size**	**Receiver Window Size**
Go-Back-N ARQ	Less than 2m	1
Selective Repeat ARQ	2m-1	2m-1

Key Point: The different window sizes in Go-Back-N and Selective Repeat ARQ represent a trade-off between simplicity (Go-Back-N) and efficiency (Selective Repeat) in handling errors and flow control in data link layer protocols.

Additional Notes

Data Link Control and Services: Framing,FlowControl,Connectionless and Connection-Oriented.

Notes on Data Link Control and Services

The sources provide insights into several critical aspects of data link control and services, including framing, flow control, and the distinction between connectionless and connection-oriented services.

Framing: Organizing Bits into Manageable Units

The data link layer is responsible for encapsulating the raw bits received from the physical layer into manageable units called frames. This process, known as **framing**, adds structure to the bit stream, demarcating the beginning and end of data units transmitted between adjacent nodes.

Importance of Framing:

- **Data Unit Separation:** Framing distinguishes individual data units, much like envelopes separate letters in the postal system.
- **Addressing and Control:** Frames incorporate header fields for addresses (sender and receiver) and control information. This enables routing and management of data flow across the network.
- **Error Detection:** Frames often include trailer fields containing error-checking mechanisms, such as the Frame Check Sequence (FCS) used in HDLC.

Types of Framing:

- **Fixed-Size Framing:** Frames maintain a constant size, as seen in Asynchronous Transfer Mode (ATM) networks where data units are called "cells". The fixed size eliminates the need for explicit boundary markers within the data stream.
- **Variable-Size Framing:** Frame sizes can vary. Two common approaches are:
    - **Character-Oriented Framing:** Relies on special characters, like the ASCII "DLE" (Data Link Escape) character, to mark frame boundaries.
    - **Bit-Oriented Framing:** Employs specific bit patterns, such as the flag sequence "01111110" in HDLC, to delineate frames. **Bit stuffing** is used to prevent this flag sequence from appearing within the data itself, ensuring accurate frame delineation.

Framing in the IEEE 802 Standard:

The IEEE 802 standard divides the data link layer into two sublayers: Logical Link Control (LLC) and Media Access Control (MAC). While the LLC handles flow and error control, framing is managed by both the LLC and MAC sublayers.

Flow Control: Preventing Data Overwhelm

Flow control mechanisms, a crucial aspect of data link control, regulate the rate at which a sender transmits data to a receiver. This prevents the receiver from being overwhelmed by data arriving faster than it can process and buffer.

Necessity of Flow Control:

- **Receiver Processing Speed:** Receivers have finite processing capabilities.
- **Buffer Limitations:** Receivers allocate buffers (memory) to temporarily store incoming data. Excessive data rates can lead to buffer overflow and data loss.

Flow Control Mechanisms:

The sources do not detail specific flow control techniques employed at the data link layer. However, they mention the use of acknowledgments (ACK frames) and negative acknowledgments (NAK frames) as part of error control mechanisms like ARQ (Automatic Repeat Request). These acknowledgments inherently contribute to flow control by informing the sender about the receiver's readiness to accept more data.

Connectionless and Connection-Oriented Services

The sources primarily focus on connection-oriented services at the data link layer, exemplified by protocols like HDLC and PPP. However, they also discuss connectionless services in the context of higher layers, notably the network layer.

Connectionless Services:

- **No Connection Setup:** Devices transmit data without prior connection establishment.
- **Independent Datagrams:** Data units (datagrams) are treated independently, each carrying complete addressing information.
- **Example:** The Internet Protocol (IP) provides a connectionless service at the network layer.

Connection-Oriented Services:

- **Connection Establishment:** A dedicated connection is established between sender and receiver before data exchange.
- **Sequenced Data Transfer:** Data units are transmitted and received in a specific order, often using sequence numbers.

- **Connection Teardown:** The connection is terminated after data transfer completes.
- **Example:** Transmission Control Protocol (TCP) offers a connection-oriented service at the transport layer. At the data link layer, protocols like HDLC and PPP operate in a connection-oriented manner.

Distinction in Terminology:

The sources use "connection service" for network-layer services and "connection-oriented service" for transport-layer services. While the terms might appear subtly different, they conceptually represent the same principle of establishing a connection before data transfer. The distinction likely arises from the different layers at which these services operate and the specific mechanisms they employ.

write notes on Two categories of links ,Two sublayers , Link Layer addressing: Three types of addresses, Address Resolution Protocol(ARP)

Notes on Link Layer Concepts

Two Categories of Links

The sources discuss two fundamental categories of network links:

1. **Point-to-Point Links:** These links establish a dedicated connection between a single sender and a single receiver. Examples of point-to-point links include connections between two routers over a long-distance link or between a user's computer and a nearby Ethernet switch. The Point-to-Point Protocol (PPP) is a widely used link-layer protocol designed for point-to-point links.
2. **Broadcast Links:** In contrast to point-to-point links, broadcast links facilitate communication among multiple sending and receiving nodes sharing a common transmission medium. Ethernet and wireless LANs are prominent examples of broadcast link-layer technologies. To manage access to the shared channel and prevent data collisions, broadcast links employ **medium access protocols**. These protocols can be centralized, with a controller dictating

transmission slots, or decentralized, where hosts coordinate among themselves.

Two Sublayers of the Data Link Layer

The sources describe how the IEEE 802 standard divides the data link layer into two sublayers:

1. **Logical Link Control (LLC):** Situated above the MAC sublayer, the LLC sublayer primarily handles framing, flow control, and error control. In essence, it enhances the raw transmission capabilities of the physical layer into a more reliable link for the network layer.
    - **Framing:** While framing is a shared responsibility between the LLC and MAC sublayers, the LLC contributes to framing by defining the format and structure of data units, ensuring they are assembled and disassembled correctly for transmission and reception.
    - **Flow Control:** While the sources don't explicitly detail flow control mechanisms at the data link layer, they allude to the role of acknowledgments (ACK frames) and negative acknowledgments (NAK frames) in protocols like Automatic Repeat Request (ARQ). These acknowledgments inherently regulate data flow by indicating the receiver's readiness to accept more data.
    - **Error Control:** The LLC sublayer employs error control techniques to ensure reliable data delivery across the link. It uses mechanisms like ARQ to manage retransmissions of lost or corrupted frames, ensuring data integrity.
2. **Media Access Control (MAC):** This sublayer governs access to the shared transmission medium, particularly crucial for broadcast links like Ethernet and wireless LANs.
    - **Multiple Access Protocols:** The MAC sublayer implements various multiple access protocols to arbitrate channel access among competing nodes. These protocols, as categorized in the sources, include:

        - **Random Access:** Nodes attempt transmission randomly, contending for the channel. Examples include ALOHA and CSMA/CD.
        - **Controlled Access:** Access to the medium is regulated through mechanisms like polling or token passing.
        - **Channelization:** The transmission medium is divided into channels, either by time (Time Division Multiple Access - TDMA), frequency (Frequency Division Multiple Access - FDMA), or code (Code Division Multiple Access - CDMA).
    - **Framing:** In addition to its role in media access control, the MAC sublayer also contributes to framing. It defines specific fields within the frame header related to addressing and media access control.

Link Layer Addressing: Three Types of Addresses

The sources highlight the importance of addressing at the link layer, particularly in the context of LANs and WANs. They discuss three primary types of addresses relevant at this layer:

1. **Physical Addresses (MAC Addresses):** These addresses, also known as link addresses, uniquely identify devices within a network segment, such as a LAN or WAN. Each network interface card (NIC) has a globally unique MAC address, usually assigned by the manufacturer.
    - **Authority:** MAC addresses hold authority within the local network, ensuring frames are delivered to the correct destination on that segment.
    - **Format and Size:** The format and size of MAC addresses can vary depending on the network technology. For example, Ethernet uses a 6-byte (48-bit) MAC address, typically represented in hexadecimal notation.
2. **Logical Addresses (IP Addresses):** While MAC addresses function locally, logical addresses, such as IP addresses, provide a global addressing scheme for devices on the Internet.

  - **Global Identification:** Logical addresses uniquely identify a host or router on the Internet, enabling communication across different networks.
  - **Hierarchical Structure:** IP addresses have a hierarchical structure, separating the network portion (network ID) from the host portion (host ID). This facilitates efficient routing and address management.

3. **Port Addresses:** Operating at a higher level than MAC and IP addresses, port addresses identify specific processes or applications running on a host.
   - **Process Differentiation:** When a host receives data, the port address within the transport-layer header directs the data to the appropriate application or service.
   - **Well-Known Ports:** Certain port numbers are standardized for specific services, such as port 80 for HTTP (web traffic) and port 25 for SMTP (email).

Address Resolution Protocol (ARP)

The sources emphasize the need for address translation between network-layer addresses (e.g., IP addresses) and link-layer addresses (e.g., MAC addresses). This translation is the primary function of the Address Resolution Protocol (ARP).

Purpose of ARP:

- **MAC Address Resolution:** When a device needs to send data to another device on the same network segment, it needs to determine the destination's MAC address. If this information is not already cached in the ARP table, the sending device initiates an ARP request.
- **ARP Request:** The ARP request is broadcast to all devices on the local network segment. The device with the matching IP address responds with its MAC address.
- **ARP Reply:** The sender caches the received MAC address in its ARP table for future use.

Example:

Consider a scenario where Host A (IP: 192.168.1.10, MAC: AA-BB-CC-DD-EE-FF) wants to send data to Host B (IP: 192.168.1.20) on the same Ethernet LAN.

1. **ARP Table Check:** Host A first checks its ARP table for an entry corresponding to Host B's IP address. If an entry exists, Host A retrieves the associated MAC address.
2. **ARP Request Broadcast:** If Host B's MAC address is not in the ARP table, Host A constructs an ARP request packet, including its own IP and MAC addresses, as well as the target IP address (192.168.1.20). This ARP request is encapsulated within an Ethernet frame and broadcast to all devices on the LAN.
3. **ARP Reply:** Host B, upon receiving the broadcast ARP request, recognizes its IP address and responds with an ARP reply. This reply includes Host B's MAC address.
4. **ARP Table Update:** Host A receives the ARP reply from Host B, updates its ARP table with the mapping between Host B's IP and MAC addresses, and then transmits the data frame using the resolved MAC address.

ARP: A Boundary-Spanning Protocol:

The sources categorize ARP as a protocol that blurs the lines between the link layer and the network layer. While an ARP packet is encapsulated within a link-layer frame, suggesting it resides above the link layer, its dependence on both MAC addresses (link layer) and IP addresses (network layer) makes its precise classification ambiguous.

Discuss different methods of random-access protocols used for broadcasting the frames in computer networks with suitable examples.

Random access protocols are used to regulate transmission into a shared broadcast channel in computer networks. These protocols are needed in settings such as wired and wireless access networks, and satellite networks. **There are three main categories of random access protocols:**

ALOHA, Carrier Sense Multiple Access (CSMA), and taking-turns protocols.

ALOHA

The earliest random access method is ALOHA, developed at the University of Hawaii. With ALOHA, a station sends a frame whenever it has one to send. Since only one channel is shared, there is a possibility of **collision** between frames from different stations. The original ALOHA protocol is called **Pure ALOHA**. The possibility of collisions in Pure ALOHA is illustrated in Figure 12.3.

Slotted ALOHA improved upon Pure ALOHA by dividing time into equal slots, and requiring that a station completely send its frame within one slot. This method increases throughput, but collisions are still possible. To resolve collisions, each station waits a random amount of time before retransmitting the frame.

CSMA

In CSMA protocols, stations **"listen" to the channel before transmitting**. This "listening" is known as **carrier sensing**. If a station senses the channel is busy, it refrains from transmitting.

CSMA/CD

CSMA/CD (Carrier Sense Multiple Access with Collision Detection) is a widely deployed CSMA protocol. In CSMA/CD, collisions are still possible because of propagation delay. After a collision, stations wait a random amount of time before attempting to retransmit.

CSMA/CA

Wireless LANs use CSMA/CA (Carrier Sense Multiple Access With Collision Avoidance). CSMA/CA uses a brief transmission from the receiver to ensure both the sender and receiver know a transmission is beginning. This helps to avoid collisions.

Taking-Turns Protocols

Taking-turns protocols allow multiple nodes to share a channel by taking turns transmitting. In these protocols, only one station can transmit at a time. This avoids collisions.

Polling

In polling, a master node polls each station in turn, giving each a chance to transmit. This is an example of a **controlled access** protocol, which is a modified form of a multiplexing technique. This method can achieve high efficiency, but it introduces polling delay and is susceptible to master node failure. 802.15 and Bluetooth are examples of polling protocols.

Token Passing

Token passing allows stations to transmit only when they hold a special packet called a token. This protocol is often used with a ring topology, in which the token circulates among the stations. This method ensures fair access to the channel. If the token is lost, a recovery procedure must be used to get it back in circulation.

Examples of Random Access Protocols:

- **Ethernet:** A widely used LAN technology that uses CSMA/CD to control access to a shared cable.
- **Wi-Fi:** A wireless LAN technology that uses CSMA/CA to avoid collisions.

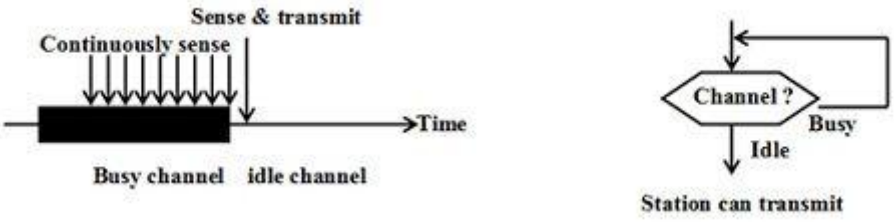
The choice of random access protocol depends on factors such as network type, data rate, and error characteristics.

List and draw flow diagrams for the persistence methods of CSMA.

CSMA (Carrier Sense Multiple Access) is a protocol that requires each station to first listen to the medium before sending data. The persistence methods define the behaviour of a station when the channel is found to be busy. There are three persistence methods:

1-Persistent

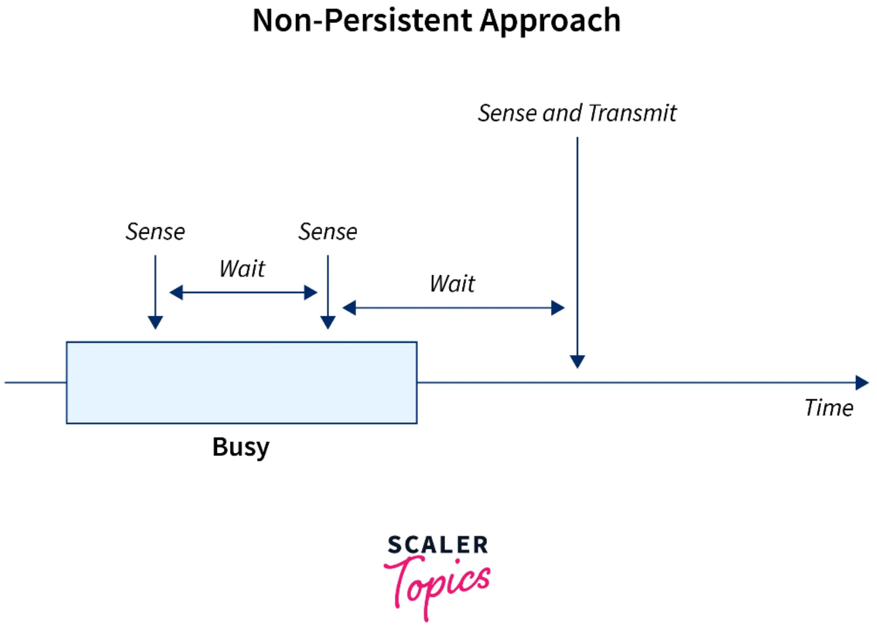
The **1-persistent method** is the simplest. In this method, if the station finds the line idle, it sends its frame immediately. If the line is busy, the station continues to listen until it becomes idle, and then it transmits the frame. If two or more stations find the channel idle at the same time, their frames will collide. Ethernet uses the 1-persistent method. The flow diagram for the 1-persistent method is shown below:



1-persistent CSMA

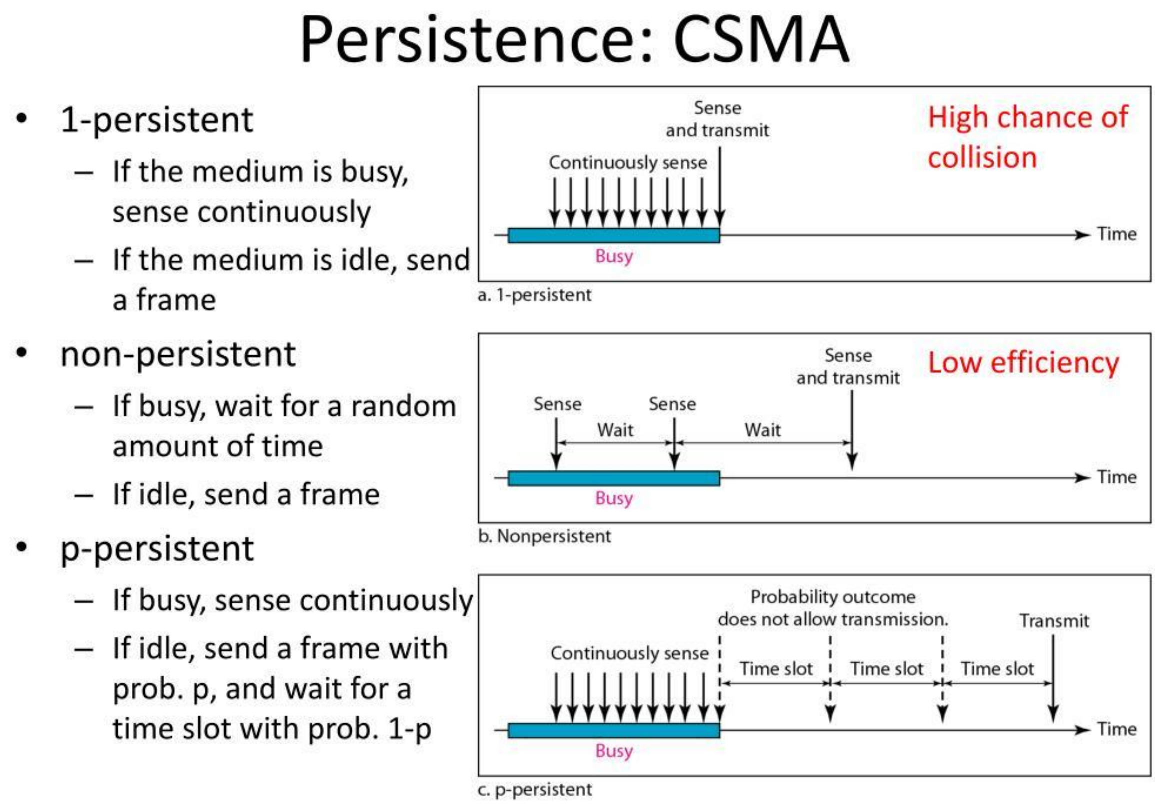
Non-Persistent

In the **non-persistent method**, if the station finds the line busy, it waits a random amount of time and then senses the line again. This method reduces the chance of collisions but can lead to inefficiency as the medium may be idle while stations wait to transmit. The flow diagram for the non-persistent method is shown below:



p-Persistent

The **p-persistent method** is a compromise between the 1-persistent and non-persistent methods. This method is used if the channel is slotted. If the station finds the line idle, it transmits with probability *p*. With probability *q* = 1 - *p*, the station waits for the beginning of the next time slot and checks the line again. If the line is idle, the process repeats. If the line is busy, the station acts as though a collision has occurred and waits a random amount of time before checking the line again. The flow diagram for the p-persistent method is shown below:



Persistence: CSMA

- 1-persistent
  - If the medium is busy, sense continuously
  - If the medium is idle, send a frame
- non-persistent
  - If busy, wait for a random amount of time
  - If idle, send a frame
- p-persistent
  - If busy, sense continuously
  - If idle, send a frame with prob. p, and wait for a time slot with prob. 1-p

a. 1-persistent

b. Nonpersistent

c. p-persistent