CBCS SCHEME

USN | | | | | | | | | | |

21CS52

Fifth Semester B.E. Degree Examination, Dec.2023/Jan.2024
Computer Networks

Time: 3 hrs. Max. Marks: 100

Note: Answer any FIVE full questions, choosing ONE full question from each module.

Module-1

1 a. Define Computer Networks. Explain local area network in detail with a neat diagram. **(06 Marks)**

b. Explain MAN with a neat labelled diagram. **(06 Marks)**

c. List and explain design issues for layer. **(08 Marks)**

OR

2 a. What are guided transmission media? Explain twisted pair cable in detail. **(06 Marks)**

b. Explain TCP/IP reference model with a neat labelled diagram. **(10 Marks)**

c. Briefly discuss virtual private networks. **(04 Marks)**

Module-2

3 a. List and explain any two data link layer design issues. **(10 Marks)**

b. A bit stream transmitted using standard CRC method. The generator polynomial is $X^3 + 1$.

i) What is actual bit string transmitted

ii) Suppose 3rd bit from the left is inverted during transmission, how will receiver detect this error? **(10 Marks)**

OR

4 a. Explain Go-Back-N protocol working. **(10 Marks)**

b. Briefly explain static channel and dynamic channel allocation problem. **(10 Marks)**

Module-3

5 a. Write an Dijkstra's algorithm to compute shortest path through graph. Explain with example. **(10 Marks)**

b. Illustrate working of OSPF and BGP. **(10 Marks)**

OR

6 a. What is congestion control? List and explain various approaches to congestion control. **(12 Marks)**

b. What is packet scheduling algorithm? Explain FIFO algorithm. **(08 Marks)**

Module-4

7 a. Write a program for congestion control using leaky bucket algorithm. **(10 Marks)**

b. Briefly explain about transport service primitives. **(10 Marks)**

OR

8 a. With a neat labelled diagram, explain TCP segment structure. **(10 Marks)**

b. Explain TCP connection management with TCP connection management FSM diagram. **(10 Marks)**

Module-5

9 a. Explain client/server and P-P architecture with a neat labelled diagram. (10 Marks)

b. Explain use and server interaction with a neat diagram. (10 Marks)

OR

10 a. Explain persistant and non persistant http in details. (10 Marks)

b. Write notes on:

(i) E-mail in the internet

(ii) Distributed DNS architecture (10 Marks)

* * * * *

1.a Define computer networks. Explain local area network in detail with a neat diagram.

A **computer network** is a collection of interconnected devices such as computers, servers, and other hardware, enabling communication and resource sharing among them. These networks facilitate the exchange of data, access to shared resources like printers and storage devices, and enable applications such as email, internet browsing, and file transfer.

Types of Computer Networks

1. **LAN (Local Area Network)**
2. **MAN (Metropolitan Area Network)**
3. **WAN (Wide Area Network)**
4. **PAN (Personal Area Network)**
5. **SAN (Storage Area Network)**
6. **CAN (Campus Area Network)**
7. **VPN (Virtual Private Network)**

Detailed Explanation of LAN (Local Area Network)

Local Area Network (LAN) is a network that connects devices within a limited geographic area, such as a home, school, office building, or closely positioned group of buildings. It is typically used for sharing resources like files, printers, and internet connections among the connected devices.

Key Characteristics of LAN:

1. **Geographical Range**: Typically limited to a small area such as a single building or a group of buildings.
2. **High Data Transfer Rates**: Provides high-speed data transfer rates, usually ranging from 10 Mbps to 10 Gbps.
3. **Ownership**: Usually owned, controlled, and managed by a single person or organization.
4. **Connectivity**: Devices are connected using cables (Ethernet) or wirelessly (Wi-Fi).
5. **Components**: Includes routers, switches, hubs, network interface cards (NICs), and cables.

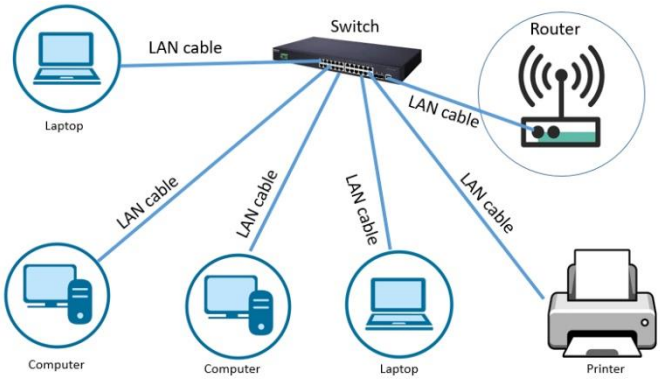
Components of a LAN:

1. **Router**: Connects the LAN to the internet or other networks.
2. **Switch**: Central device that connects multiple devices within the LAN and forwards data based on MAC addresses.
3. **Hub**: Similar to a switch but broadcasts data to all connected devices.
4. **Network Interface Card (NIC)**: Hardware component in each device that allows it to connect to the LAN.
5. **Cables**: Physical medium used to connect devices (e.g., Ethernet cables).
6. **Access Points**: Devices that provide wireless connectivity within the LAN.

Diagram of a Local Area Network (LAN)

Here is a description for the diagram:

1. **Central Router**: Positioned in the center, connecting to the internet.
2. **Switch**: Connected to the router, with multiple devices linked to the switch.
3. **Devices**: Various devices such as computers, printers, and servers connected to the switch.
4. **Wireless Access Point**: Connected to the switch, providing wireless connectivity to laptops and mobile devices.
5. **Internet Connection**: Indicated by a cloud icon, connected to the router.

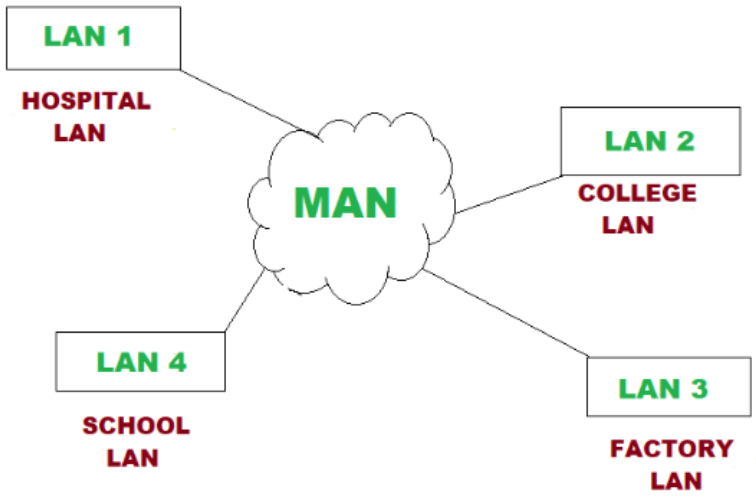


Local Area Network

b. Explain MAN with a neat labeled diagram.

A Metropolitan Area Network (MAN) is a type of computer network that spans over a metropolitan area, typically a city. It provides high-speed data communication services such as video, audio, and data transfer between multiple LANs (Local Area Networks) and WANs (Wide Area Networks). The main purpose of a MAN is to connect different LANs in a city to share resources and exchange data, as well as to provide internet access to users. A MAN typically covers a geographic area of several kilometers and is larger than a LAN but smaller than a WAN.

MAN stands for **Metropolitan Area Network**. It is a computer network that connects number of LANs to form larger network, so that the computer resources can be shared. This type of network covers larger area than a LAN but smaller than the area covered by a WAN which is designed to extend over the entire city.



c. List and explain design issues for layer.

In computer networks, the architecture is typically divided into layers, each with specific functions and design issues. The most commonly referenced model is the OSI (Open Systems Interconnection) model, which has seven layers. Here, we'll explore the design issues associated with each of these layers.

OSI Model Layers and Their Design Issues

1. **Physical Layer**
2. **Data Link Layer**
3. **Network Layer**
4. **Transport Layer**
5. **Session Layer**
6. **Presentation Layer**
7. **Application Layer**

1. Physical Layer

Function: Transmits raw bit streams over a physical medium.

Design Issues:

- **Bit Synchronization**: Ensuring the sender and receiver are synchronized at the bit level.
- **Transmission Medium**: Selection of the appropriate physical medium (e.g., copper wires, fiber optics, wireless).
- **Signal Encoding**: Choosing the method to represent bits (e.g., voltage levels, frequencies).
- **Physical Topology**: Designing the layout of the network (e.g., star, ring, bus).
- **Data Rate**: Determining the speed at which data is transmitted.

2. Data Link Layer

Function: Provides node-to-node data transfer and error detection/correction.

Design Issues:

- **Framing**: Dividing the bit stream into frames.
- **Error Control**: Detecting and correcting errors in transmitted frames.
- **Flow Control**: Managing the rate of data transmission between sender and receiver.
- **MAC (Media Access Control)**: Determining how multiple devices share the same medium (e.g., CSMA/CD in Ethernet).
- **Addressing**: Assigning physical addresses to devices.

3. Network Layer

Function: Handles routing of data between devices across different networks.

Design Issues:

- **Routing**: Determining the optimal path for data to travel from source to destination.
- **Logical Addressing**: Assigning IP addresses to devices.
- **Packet Forwarding**: Moving packets from the source to the destination.
- **Congestion Control**: Managing network congestion to ensure efficient data transfer.
- **Inter-Networking**: Connecting different types of networks and managing compatibility.

4. Transport Layer

Function: Provides reliable data transfer between end systems.

Design Issues:

- **Segmentation and Reassembly**: Dividing data into segments and reassembling them at the destination.
- **Error Control**: Ensuring all segments are delivered without errors.
- **Flow Control**: Preventing sender from overwhelming the receiver.
- **Connection Management**: Establishing, maintaining, and terminating connections (e.g., TCP).
- **Multiplexing**: Allowing multiple applications to use the network simultaneously.

5. Session Layer

Function: Manages sessions between applications.

Design Issues:

- **Session Establishment**: Initiating and maintaining sessions.

- **Synchronization**: Managing dialog control and synchronization points within a session.
- **Session Termination**: Gracefully ending sessions.
- **Dialog Control**: Determining who can send data at a given time.

6. Presentation Layer

Function: Translates data between the application and the network format.

Design Issues:

- **Data Translation**: Converting data formats between sender and receiver (e.g., ASCII to EBCDIC).
- **Data Encryption**: Encrypting data for secure transmission.
- **Data Compression**: Reducing the size of data to be transmitted.

7. Application Layer

Function: Provides network services directly to applications.

Design Issues:

- **Service Interface**: Providing standard interfaces for applications to access network services.
- **Application Protocols**: Implementing protocols for specific applications (e.g., HTTP, FTP, SMTP).
- **User Authentication**: Verifying the identity of users accessing the network services.
- **Data Integrity**: Ensuring data is not altered during transmission.

2.a. What are guided transmission media? Explain twisted pair cable in detail.

Guided transmission media are physical pathways that guide the transmission of data signals. These media provide a conduit from one device to another and include cables such as twisted pair cables, coaxial cables, and fiber optic cables.

Types of Guided Transmission Media

1. **Twisted Pair Cable**
2. **Coaxial Cable**
3. **Fiber Optic Cable**

Twisted Pair Cable

Twisted pair cable is one of the most commonly used types of guided transmission media, especially in local area networks (LANs) and telephone systems.

Structure

A twisted pair cable consists of pairs of insulated copper wires twisted together. The twisting helps to reduce electromagnetic interference (EMI) and crosstalk between adjacent pairs. There are two main types of twisted pair cables:

1. **Unshielded Twisted Pair (UTP)**
2. **Shielded Twisted Pair (STP)**

Types of Twisted Pair Cables

1. **Unshielded Twisted Pair (UTP)**:
    - **Cat 3**: Used in telephone networks; supports up to 10 Mbps.
    - **Cat 5**: Used in LANs; supports up to 100 Mbps.
    - **Cat 5e**: Enhanced version of Cat 5; supports up to 1 Gbps.
    - **Cat 6**: Used in high-speed networks; supports up to 10 Gbps over short distances.
    - **Cat 6a**: Augmented Cat 6; supports up to 10 Gbps over longer distances.
    - **Cat 7**: Shielded variant with improved performance; supports up to 10 Gbps.
2. **Shielded Twisted Pair (STP)**:
    - Contains an additional shielding material to reduce interference further.
    - Used in environments with high EMI, such as industrial settings.

Detailed Explanation of Twisted Pair Cable

1. Construction and Design:

- **Pairs of Wires**: Each pair consists of two copper wires twisted around each other.
- **Insulation**: Each wire is coated with an insulating material.
- **Shielding (in STP)**: Additional metallic shielding wraps around pairs to prevent interference.

2. Working Principle:

- **Twisting**: The twisting of pairs helps cancel out electromagnetic interference from external sources and reduces crosstalk between pairs in the same cable.
- **Differential Signaling**: Signals are transmitted in differential mode, where one wire carries the signal and the other carries the inverted signal. This helps in rejecting noise.

3. Categories of UTP Cables:

- **Cat 3**: Suitable for voice communication and older data networks.
- **Cat 5**: Commonly used for Ethernet networks; supports 100 Mbps.
- **Cat 5e**: Enhanced version of Cat 5; reduces crosstalk and supports 1 Gbps.
- **Cat 6**: Used for high-speed networks; supports up to 10 Gbps for shorter distances.
- **Cat 6a**: Supports 10 Gbps over longer distances than Cat 6.
- **Cat 7**: Provides higher shielding and supports up to 10 Gbps.

4. Applications:

- **Telephone Networks**: UTP cables are extensively used in telephone systems.
- **Local Area Networks (LANs)**: Commonly used in office and home networking.
- **Data Transmission**: Used for internet connectivity, data transfer between computers, and connecting network devices.

5. Advantages:

- **Cost-Effective**: Relatively inexpensive compared to other guided media.
- **Easy to Install**: Flexible and easy to work with, making installation straightforward.
- **Widely Available**: Readily available and supported by a wide range of networking equipment.

6. Disadvantages:

- **Distance Limitation**: Limited to shorter distances compared to fiber optic cables.
- **Susceptible to Interference**: While twisting reduces interference, UTP is still more susceptible compared to coaxial and fiber optic cables.
- **Bandwidth Limitations**: Higher categories support greater bandwidth, but still less than fiber optics.

b. Explain TCP/IP reference model with a neat labeled diagram.

The TCP/IP reference model, also known as the Internet protocol suite, is a set of communication protocols used for the internet and similar networks. It was developed by the United States Department of Defense to enable reliable communication over a network of arbitrary design. The model is divided into four layers:

1. **Application Layer**
2. **Transport Layer**
3. **Internet Layer**
4. **Network Interface Layer**

Layers of the TCP/IP Model

1. Application Layer

Function: Provides various network services to applications. It supports processes such as email, file transfer, and web browsing.

Key Protocols:

- **HTTP (Hypertext Transfer Protocol)**: Used for web browsing.
- **FTP (File Transfer Protocol)**: Used for transferring files.

- **SMTP (Simple Mail Transfer Protocol)**: Used for email transmission.
- **DNS (Domain Name System)**: Resolves domain names to IP addresses.

2. Transport Layer

Function: Provides end-to-end communication services for applications. It ensures complete data transfer.

Key Protocols:

- **TCP (Transmission Control Protocol)**: Provides reliable, ordered, and error-checked delivery of data.
- **UDP (User Datagram Protocol)**: Provides a connectionless, lightweight, and faster transmission.

3. Internet Layer

Function: Handles the movement of packets around the network. It defines the logical addressing and routing of packets.

Key Protocols:

- **IP (Internet Protocol)**: Provides logical addressing (IP addresses) and routing of packets.
- **ICMP (Internet Control Message Protocol)**: Used for error messages and operational information.
- **ARP (Address Resolution Protocol)**: Resolves IP addresses to MAC (physical) addresses.
- **RARP (Reverse Address Resolution Protocol)**: Resolves MAC addresses to IP addresses.

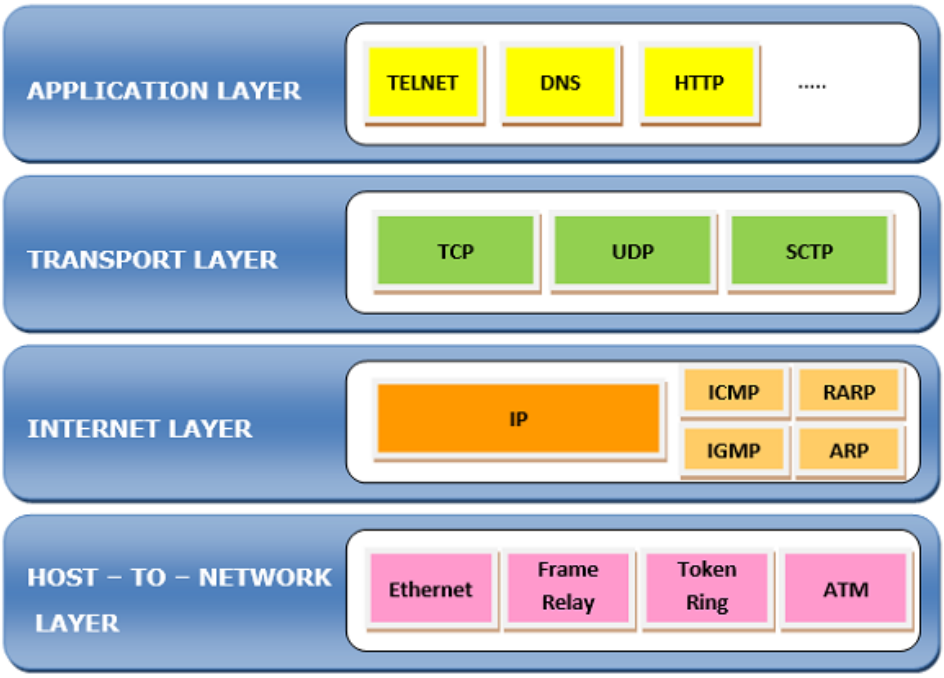
4. Network Interface Layer (Link Layer)

Function: Deals with the physical connection to the network. It includes the hardware and device drivers that interact with the physical medium.

Key Protocols and Technologies:

- **Ethernet**: A common LAN technology.
- **Wi-Fi**: Wireless LAN technology.

- **Frame Relay**: Used for wide area networks (WANs).
- **PPP (Point-to-Point Protocol)**: Used for direct connections between two nodes.



c. Briefly discuss Virtual Private Networks.

A **Virtual Private Network (VPN)** is a technology that creates a secure and encrypted connection over a less secure network, such as the internet. VPNs are commonly used to ensure privacy and security for data transmitted over public or shared networks.

Key Points about VPNs:

1. **Secure Connection**:
   - VPNs establish a secure tunnel between the user's device and the VPN server. This tunnel encrypts the data, making it unreadable to anyone who might intercept it.
2. **Privacy and Anonymity**:
   - By masking the user's IP address with that of the VPN server, VPNs provide a level of anonymity. This makes it difficult for websites and online services to track the user's activities and location.
3. **Remote Access**:
   - VPNs allow remote users to securely connect to a private network, such as a corporate network, from anywhere in the world. This is particularly useful for employees who need to access company resources while working remotely.
4. **Bypass Geographical Restrictions**:

    - VPNs can be used to bypass geographical restrictions and censorship by routing the connection through a server in a different location. This allows users to access content that may be restricted in their country.

Common Uses of VPNs:

- **Corporate Use**: Companies use VPNs to enable remote workers to securely access the company's internal network.
- **Personal Use**: Individuals use VPNs to protect their privacy, secure their internet connection, and access region-locked content.
- **Enhanced Security**: VPNs provide an additional layer of security when using public Wi-Fi networks, protecting against potential threats such as hackers and identity theft.

3.a. List and explain any two data link layer design issues.

The Data Link Layer is the second layer in the OSI model, responsible for node-to-node data transfer, error detection and correction, and flow control. Here are two significant design issues at the Data Link Layer:

1. **Framing**
2. **Error Control**

1. Framing

Definition: Framing is the process of breaking down the continuous stream of bits from the physical layer into manageable data units called frames.

Key Points:

- **Purpose**: The main purpose of framing is to ensure that the sender and receiver can distinguish one frame from another. This helps in the proper synchronization and error detection of data packets.
- **Frame Structure**: Each frame typically consists of a header, payload, and trailer. The header and trailer contain control information such as addresses, error-checking codes, and other metadata.
- **Types of Framing Techniques**:
    - **Character Count**: Uses a field in the header to specify the number of characters in the frame.
    - **Byte Stuffing**: Special characters are used to indicate the start and end of a frame. If these special characters appear in the data, they are "stuffed" with an escape character to differentiate them.
    - **Bit Stuffing**: Uses a pattern of bits to indicate frame boundaries. If the same pattern appears in the data, extra bits are added to avoid confusion.
- **Challenges**:
    - **Synchronization**: Ensuring that the receiver correctly identifies the start and end of each frame.

    - **Overhead**: Adding headers and trailers to frames increases the amount of data that needs to be transmitted.
    - **Efficiency**: Balancing the frame size to optimize the trade-off between overhead and the likelihood of errors within a frame.

Importance:

- Framing is crucial for data integrity and efficient communication. It ensures that data is correctly interpreted and processed by the receiving node.

2. Error Control

Definition: Error control is the mechanism for detecting and correcting errors that occur during the transmission of data frames.

Key Points:

- **Purpose**: The main purpose of error control is to ensure reliable communication by detecting and correcting errors in transmitted frames.
- **Error Detection Techniques**:
    - **Parity Check**: Adds a parity bit to the frame to make the number of set bits either even (even parity) or odd (odd parity).
    - **Cyclic Redundancy Check (CRC)**: Uses polynomial division to generate a checksum that is appended to the frame. The receiver performs the same division and compares the result with the checksum.
    - **Checksum**: Calculates a sum of all data bytes and appends it to the frame. The receiver calculates the sum again and compares it with the received checksum.
- **Error Correction Techniques**:
    - **Automatic Repeat Request (ARQ)**: Uses acknowledgments and timeouts to ensure frames are received correctly. If an error is detected, the frame is retransmitted. Common ARQ protocols include Stop-and-Wait ARQ, Go-Back-N ARQ, and Selective Repeat ARQ.
    - **Forward Error Correction (FEC)**: Adds redundant data to the frame, allowing the receiver to detect and correct errors without needing retransmission.
- **Challenges**:
    - **Latency**: Error detection and correction mechanisms can introduce delays, especially in ARQ protocols where retransmissions are involved.
    - **Bandwidth**: Additional bits for error detection and correction increase the amount of data transmitted, affecting bandwidth efficiency.
    - **Complexity**: Implementing robust error control mechanisms increases the complexity of both hardware and software in network devices.

Importance:

- Error control is essential for maintaining the reliability and integrity of data communication. It ensures that errors caused by noise, signal degradation, or other factors are detected and corrected, thereby preventing data corruption.

4.a. Explain go back n protocol working.

The Go-Back-N Automatic Repeat Request (ARQ) protocol is a sliding window protocol used for error control in data transmission. It ensures reliable delivery of frames over a potentially unreliable communication channel by handling errors and lost frames.

Key Concepts

1. **Sliding Window**:
    - The sender maintains a window of frames that can be sent before requiring an acknowledgment (ACK) from the receiver.
    - The receiver has a window size of one, meaning it can only accept frames in sequence.
2. **Sequence Numbers**:
    - Each frame is assigned a unique sequence number.
    - The sequence number space must be larger than the window size to avoid ambiguity.
3. **Acknowledgments**:
    - The receiver sends an ACK for the last correctly received frame.
    - Cumulative ACKs indicate that all previous frames have been received correctly.

Working of Go-Back-N Protocol

1. **Sender Side**:
    - The sender can send multiple frames (up to the window size) without waiting for an acknowledgment.
    - It maintains a timer for the oldest unacknowledged frame.
    - If the timer expires, the sender retransmits all frames from the oldest unacknowledged frame onward.
2. **Receiver Side**:
    - The receiver accepts frames in the correct sequence.
    - If a frame is received out of order (i.e., it is not the expected frame), the receiver discards it and sends an ACK for the last correctly received frame.
    - The receiver does not buffer out-of-order frames.
3. **Error Handling**:
    - If a frame is lost or an error is detected, the receiver discards it and continues to acknowledge the last correctly received frame.
    - The sender, upon timeout, retransmits all frames from the last acknowledged frame.

Example Scenario

Let's consider an example where the window size is 4, and the sender wants to send frames 0, 1, 2, 3, 4, 5, 6, 7.

1. **Initial Transmission**:
    - The sender sends frames 0, 1, 2, 3.
    - The receiver receives frames 0, 1, 2, 3 and sends cumulative ACK for frame 3.
2. **Next Transmission**:
    - The sender slides the window and sends frames 4, 5, 6, 7.
    - The receiver receives frame 4, but frame 5 is lost or corrupted.
3. **Acknowledgment**:
    - The receiver discards frame 6 and 7 (since frame 5 was not received) and sends an ACK for frame 4.
4. **Retransmission**:
    - The sender, upon timeout or receiving the ACK for frame 4, retransmits frames 5, 6, 7.
5. **Successful Transmission**:
    - The receiver correctly receives frames 5, 6, 7 and sends cumulative ACK for frame 7.

b. Briefly explain static channel and dynamic channel allocation problem.

Static Channel Allocation

Static Channel Allocation is a method where a fixed number of channels are allocated to users or devices in a network. This method is simple and predictable but can be inefficient in environments where the demand for channel access varies.

Key Points:

1. **Fixed Assignment**: Channels are permanently assigned to specific users or devices, regardless of whether they are in use or not.
2. **Inefficiency**: If a channel is assigned to a user who is not transmitting data, that channel remains unused, leading to inefficient utilization of available bandwidth.
3. **Predictability**: This method is easy to manage and predict because channel assignments do not change.
4. **Examples**: Traditional telephone networks, where each line is a fixed channel, and Frequency Division Multiple Access (FDMA) in cellular networks, where each user gets a specific frequency band.

Dynamic Channel Allocation

Dynamic Channel Allocation is a method where channels are allocated to users or devices as needed. This approach is more efficient in environments with variable demand for channel access.

Key Points:

1. **On-Demand Assignment**: Channels are assigned to users dynamically based on current demand and availability.
2. **Efficiency**: This method optimizes the use of available bandwidth, as channels are only allocated when needed.
3. **Complexity**: Managing dynamic allocation requires more sophisticated algorithms and real-time management to avoid conflicts and ensure fair access.
4. **Examples**: Time Division Multiple Access (TDMA) and Code Division Multiple Access (CDMA) in cellular networks, where channels are assigned based on current user demand.

Comparison

1. **Utilization**:
    - **Static Allocation**: Often leads to poor utilization of channels because some channels may remain unused when not needed by their assigned users.
    - **Dynamic Allocation**: Leads to better utilization of channels as they are assigned based on demand.
2. **Complexity**:
    - **Static Allocation**: Simpler to implement and manage, with less overhead in channel management.
    - **Dynamic Allocation**: More complex to implement due to the need for real-time monitoring and allocation algorithms.
3. **Flexibility**:
    - **Static Allocation**: Less flexible, as channels are fixed and cannot adapt to changes in user demand.
    - **Dynamic Allocation**: More flexible, can adapt to varying user demand and traffic patterns.
4. **Fairness**:
    - **Static Allocation**: May be fair in environments with predictable and uniform traffic but can lead to unfairness in environments with variable demand.
    - **Dynamic Allocation**: Can ensure fair access to channels by dynamically assigning channels based on current needs.

5.a. Write an Dijkstra's algorithm algorithm to compute shortest path through graph. Explain with example.

Dijkstra's algorithm is a famous algorithm used for finding the shortest paths between nodes in a graph. It works with graphs that have non-negative weights. Here is the algorithm and a step-by-step example to illustrate how it works.

Algorithm

1. **Initialization**:
    - Set the distance to the starting node to 0 and the distance to all other nodes to infinity.
    - Set the starting node as the current node.

- Create a set of visited nodes (initially empty) and a priority queue of nodes to be processed, initialized with the starting node.

2. **Process Current Node**:
   - For the current node, consider all of its unvisited neighbors and calculate their tentative distances. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one.
   - After considering all of the neighbors, mark the current node as visited and remove it from the unvisited set.
3. **Update Priority Queue**:
   - Select the unvisited node with the smallest tentative distance and set it as the new current node.
4. **Repeat**:
   - Repeat steps 2 and 3 until all nodes have been visited or the smallest tentative distance among the unvisited nodes is infinity (which happens when there is no connection between the remaining unvisited nodes and the start node).
5. **Result**:
   - The algorithm finishes when all nodes have been visited. The shortest path to each node is determined.

Pseudo code

```
function Dijkstra(Graph, source):

    dist[source] ← 0

    for each vertex v in Graph:

        if v ≠ source:

            dist[v] ← ∞

        add v to the priority queue Q

    while Q is not empty:

        u ← vertex in Q with the smallest distance in dist[]

        remove u from Q

        for each neighbor v of u:

            alt ← dist[u] + length(u, v)

            if alt < dist[v]:

                dist[v] ← alt

                update priority queue Q with new distance for v
```

return dist[]

6.a. What is congestion control. List and explain various approaches to congestion control.

Congestion control refers to the techniques and mechanisms used to prevent or alleviate congestion in a network. Congestion occurs when the demand for network resources exceeds the available capacity, leading to packet loss, increased delays, and reduced throughput. Effective congestion control ensures efficient and reliable data transmission.

Approaches to Congestion Control

There are several approaches to congestion control, which can be broadly categorized into two types: **Open-loop Control** (preventive measures) and **Closed-loop Control** (reactive measures).

1. Open-loop Congestion Control

Open-loop control aims to prevent congestion before it happens by using policies and mechanisms that ensure the network operates within its capacity limits. Key approaches include:

- **Traffic Shaping**:
  - **Leaky Bucket Algorithm**: Enforces a constant output rate regardless of the burstiness of the incoming traffic by buffering and smoothing the traffic flow.
  - **Token Bucket Algorithm**: Allows bursty traffic up to a certain limit, providing more flexibility than the leaky bucket by using tokens to control the transmission of packets.
- **Admission Control**:
  - Prevents new traffic from entering the network if it is predicted that the additional load will lead to congestion. This approach ensures that only a manageable amount of traffic is allowed, maintaining network performance.
- **Resource Reservation**:
  - Reserves necessary resources (such as bandwidth) for certain applications or flows, ensuring that these resources are available when needed. Protocols like RSVP (Resource Reservation Protocol) are used for this purpose.

2. Closed-loop Congestion Control

Closed-loop control involves detecting congestion once it occurs and taking corrective actions to alleviate it. Key approaches include:

- **Backpressure**:
  - Nodes experiencing congestion send signals upstream to slow down or halt the transmission of packets. This prevents further congestion by controlling the traffic flow at its source.
- **Choke Packet**:

    - A congested node sends a special packet (choke packet) to the source node, instructing it to reduce its transmission rate. This directly informs the sender about the congestion, prompting it to adjust its behavior.
- **Implicit Congestion Signaling**:
    - End systems infer congestion based on packet loss, delays, or variations in round-trip time (RTT) and adjust their transmission rates accordingly. Techniques like TCP congestion control use these signals to manage congestion.
- **Explicit Congestion Notification (ECN)**:
    - Network devices mark packets to indicate congestion to the end systems, which then reduce their transmission rates. ECN allows for proactive congestion management without waiting for packet loss.

TCP Congestion Control Mechanisms

TCP (Transmission Control Protocol) uses several mechanisms to control congestion effectively:

- **Slow Start**:
    - Begins with a small congestion window (cwnd) and exponentially increases it with each acknowledgment received, until a threshold (ssthresh) is reached or packet loss occurs.
- **Congestion Avoidance**:
    - After reaching the threshold, the increase in the congestion window becomes linear, reducing the rate of growth to avoid congestion.
- **Fast Retransmit and Fast Recovery**:
    - When duplicate acknowledgments (indicating packet loss) are received, TCP retransmits the lost packet without waiting for a timeout (Fast Retransmit) and reduces the congestion window size, but not as drastically as in Slow Start (Fast Recovery).

b. What is packet scheduling algorithm? Explain FIFO algorithm.

Packet scheduling algorithms are techniques used in network devices (like routers and switches) to determine the order in which packets are transmitted. These algorithms play a crucial role in managing network traffic, ensuring efficient use of bandwidth, and providing quality of service (QoS) by prioritizing certain types of traffic.

FIFO (First In, First Out) Algorithm

The **FIFO (First In, First Out)** algorithm, also known as **First-Come, First-Served (FCFS)**, is one of the simplest and most straightforward packet scheduling algorithms.

How FIFO Works

- **Queue Management**: Packets are placed in a single queue in the order they arrive.
- **Processing Order**: The packet at the front of the queue (the one that has been in the queue the longest) is processed and transmitted first.

- **No Prioritization**: FIFO treats all packets equally, without any differentiation based on their source, destination, type, or priority.

Example of FIFO Algorithm

Consider a network router where packets arrive in the following order:

- Packet A arrives at time T1
- Packet B arrives at time T2
- Packet C arrives at time T3

The FIFO queue will handle these packets in the order of their arrival:

1. Packet A will be transmitted first.
2. Packet B will be transmitted next.
3. Packet C will be transmitted last.

Advantages of FIFO

- **Simplicity**: FIFO is easy to implement and understand, requiring minimal computational resources.
- **Fairness**: All packets are treated equally, so no packet is given preferential treatment over others.

Disadvantages of FIFO

- **Lack of Prioritization**: Critical or time-sensitive packets are not given priority, which can lead to increased latency for important traffic.
- **Potential for Long Delays**: If a large packet is at the front of the queue, it can cause delays for smaller packets behind it, leading to inefficiency in certain scenarios.

Use Cases

- **Best-Effort Networks**: FIFO is suitable for networks where all traffic is treated equally and there are no specific QoS requirements.
- **Simple Systems**: It is used in systems where simplicity and low overhead are more important than advanced traffic management features.

Comparison with Other Packet Scheduling Algorithms

- **Priority Queuing (PQ)**: Unlike FIFO, PQ assigns different priority levels to packets. High-priority packets are processed before low-priority ones.
- **Round Robin (RR)**: This algorithm processes packets from multiple queues in a cyclic order, ensuring a fair distribution of bandwidth among different traffic flows.

- **Weighted Fair Queuing (WFQ)**: WFQ assigns weights to different queues, allowing more important traffic to get a larger share of the bandwidth while still providing fair access to other traffic.

7.a. Write a program for congestion control using leaky bucket algorithm.

The Leaky Bucket algorithm is a traffic shaping algorithm that controls the data rate of a network by regulating the flow of packets. It works like a bucket with a hole at the bottom, where the water (packets) leaks out at a steady rate. If the bucket overflows due to incoming water (packets) being added too quickly, the excess water is discarded.

Here's a Python program to simulate congestion control using the Leaky Bucket algorithm:

```
import time

import random


class LeakyBucket:

    def __init__(self, bucket_size, output_rate):

        self.bucket_size = bucket_size

        self.output_rate = output_rate

        self.current_water = 0  # Current water level in the bucket


    def add_packet(self, packet_size):

        if self.current_water + packet_size > self.bucket_size:

            print(f"Packet of size {packet_size} dropped. Bucket overflow!")

        else:

            self.current_water += packet_size

            print(f"Packet of size {packet_size} added to the bucket. Current water level: {self.current_water}")


    def leak(self):
```

```
        if self.current_water > 0:
            self.current_water = max(0, self.current_water - self.output_rate)
            print(f"Leaked {self.output_rate} units. Current water level: {self.current_water}")
        else:
            print("Bucket is empty, nothing to leak.")

def simulate_leaky_bucket(bucket, packets, interval):
    for packet_size in packets:
        bucket.add_packet(packet_size)
        bucket.leak()
        time.sleep(interval)

if __name__ == "__main__":
    # Configuration
    bucket_size = 10  # The size of the bucket
    output_rate = 2   # The rate at which packets are leaked from the bucket
    interval = 1      # Interval between packet arrivals

    # Create a LeakyBucket instance
    leaky_bucket = LeakyBucket(bucket_size, output_rate)

    # Generate a list of packets with random sizes
    packets = [random.randint(1, 5) for _ in range(10)]

    print(f"Generated packets: {packets}")
```

```
# Simulate the leaky bucket process

simulate_leaky_bucket(leaky_bucket, packets, interval)
```

Explanation:

1. **LeakyBucket Class**:
    - __init__: Initializes the bucket size, output rate, and current water level.
    - add_packet: Adds a packet to the bucket. If adding the packet would overflow the bucket, the packet is dropped.
    - leak: Simulates the leaking process by reducing the current water level by the output rate.
2. **simulate_leaky_bucket Function**:
    - Simulates the process by iterating through a list of packets. Each packet is added to the bucket, and then the bucket leaks.
3. **Main Execution**:
    - Configures the bucket size, output rate, and interval.
    - Creates a LeakyBucket instance.
    - Generates a list of packets with random sizes.
    - Simulates the leaky bucket process.

b. Briefly explain about transport service primitives.

Transport Service Primitives

Transport service primitives are the basic operations or functions provided by the transport layer of a network to enable communication between applications on different devices. These primitives facilitate the establishment, maintenance, and termination of connections, as well as data transfer between end systems.

The transport layer's primary role is to provide reliable data transfer services, ensuring that data is delivered accurately and in order. Here is a brief overview of the key transport service primitives:

1. **LISTEN**:
    - **Purpose**: To prepare a server to accept incoming connection requests.
    - **Functionality**: The server application uses this primitive to indicate its willingness to accept connections from clients. It typically specifies a port number on which it will listen for incoming requests.
2. **CONNECT**:
    - **Purpose**: To establish a connection between a client and a server.
    - **Functionality**: The client application uses this primitive to initiate a connection to a listening server. It provides the server's address and port number. If the connection is successful, a communication channel is established.

3. **ACCEPT**:
    - **Purpose**: To accept an incoming connection request from a client.
    - **Functionality**: The server application uses this primitive to accept a connection request that has been received while it was listening. This primitive completes the connection establishment process and creates a new socket for communication with the client.
4. **SEND**:
    - **Purpose**: To send data from one application to another over an established connection.
    - **Functionality**: Both client and server applications use this primitive to transmit data. The data is packaged into segments and sent to the remote application.
5. **RECEIVE**:
    - **Purpose**: To receive data from the remote application.
    - **Functionality**: Both client and server applications use this primitive to read incoming data. The received data is extracted from the segments and delivered to the application.
6. **DISCONNECT**:
    - **Purpose**: To terminate an existing connection.
    - **Functionality**: Either the client or the server application can use this primitive to gracefully close a connection. This ensures that all data in transit is delivered and the connection is properly closed.

Example Workflow

Consider a simple client-server model where a client wants to communicate with a server. The sequence of transport service primitives would be as follows:

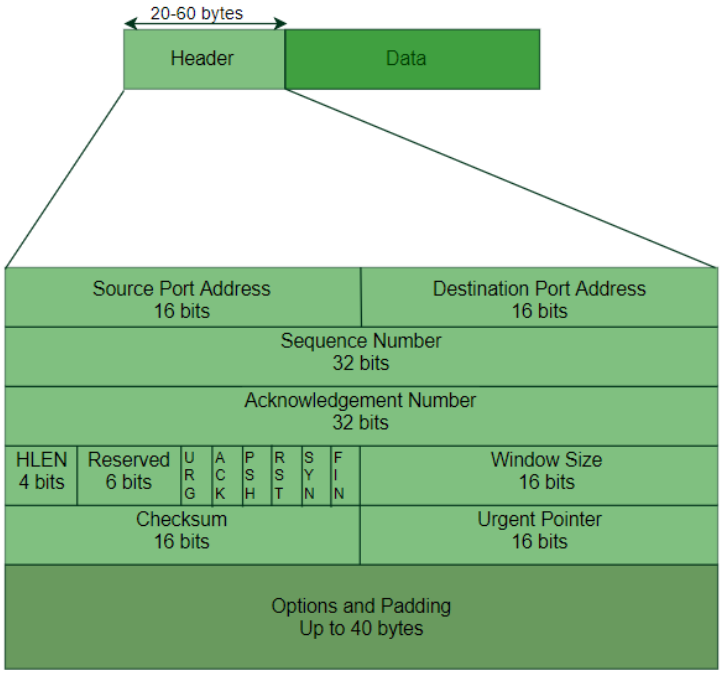
1. **Server**:
    - LISTEN(port): The server listens for incoming connections on a specific port.
2. **Client**:
    - CONNECT(server_address, port): The client requests a connection to the server's address and port.
3. **Server**:
    - ACCEPT(): The server accepts the client's connection request.
4. **Client and Server**:
    - SEND(data): The client sends data to the server.
    - RECEIVE(data): The server receives the data sent by the client.
    - SEND(data): The server sends a response to the client.
    - RECEIVE(data): The client receives the response from the server.
5. **Client or Server**:
    - DISCONNECT(): Either the client or server terminates the connection once communication is complete.

8.a. With a neat labeled diagram, explain TCP segment structure.

A TCP segment consists of a header and data. The header is composed of several fields, each with a specific purpose, essential for managing the transmission of data. Below is a detailed description of each field:

1. **Source Port (16 bits)**: Identifies the sending port.
2. **Destination Port (16 bits)**: Identifies the receiving port.
3. **Sequence Number (32 bits)**: Used to ensure data is received in order and to identify the position of the segment's data in the sequence of bytes being sent.
4. **Acknowledgment Number (32 bits)**: Indicates the next expected byte from the sender, used for acknowledgment in reliable delivery.
5. **Data Offset (4 bits)**: Specifies the size of the TCP header.
6. **Reserved (3 bits)**: Reserved for future use and should be set to zero.
7. **Flags (9 bits)**: Control flags such as URG, ACK, PSH, RST, SYN, and FIN, used to manage the state of the connection.
    - **URG**: Urgent Pointer field significant
    - **ACK**: Acknowledgment field significant
    - **PSH**: Push Function
    - **RST**: Reset the connection
    - **SYN**: Synchronize sequence numbers
    - **FIN**: No more data from sender
8. **Window Size (16 bits)**: Size of the receive window, which specifies the number of bytes the sender is willing to receive.
9. **Checksum (16 bits)**: Used for error-checking the header and data.
10. **Urgent Pointer (16 bits)**: Points to the sequence number of the byte following the urgent data.
11. **Options (Variable)**: Optional field for additional control options.
12. **Padding**: Added to ensure the header length is a multiple of 32 bits.
13. **Data**: The payload, or actual data being transmitted.

Here is the labeled diagram for a TCP segment structure:



b. Explain TCP connection management with FSM diagram.

TCP connection management involves three main phases: connection establishment, data transfer, and connection termination. These phases are governed by a finite state machine (FSM) to ensure reliable and orderly communication. Here is an explanation of each phase, followed by an FSM diagram that visualizes these transitions.

1. Connection Establishment (Three-Way Handshake)

- **SYN Sent**: The client sends a SYN (synchronize) packet to the server, indicating a request to establish a connection.
- **SYN-ACK Received**: The server responds with a SYN-ACK (synchronize-acknowledge) packet, acknowledging the client's SYN and indicating its own request to establish a connection.
- **ACK Sent**: The client sends an ACK (acknowledge) packet, acknowledging the server's SYN-ACK, and the connection is established.

2. Data Transfer

- **ESTABLISHED**: Once the connection is established, data can be transferred between the client and the server. Each side can send and receive data packets, which are acknowledged by the receiving end to ensure reliable delivery.

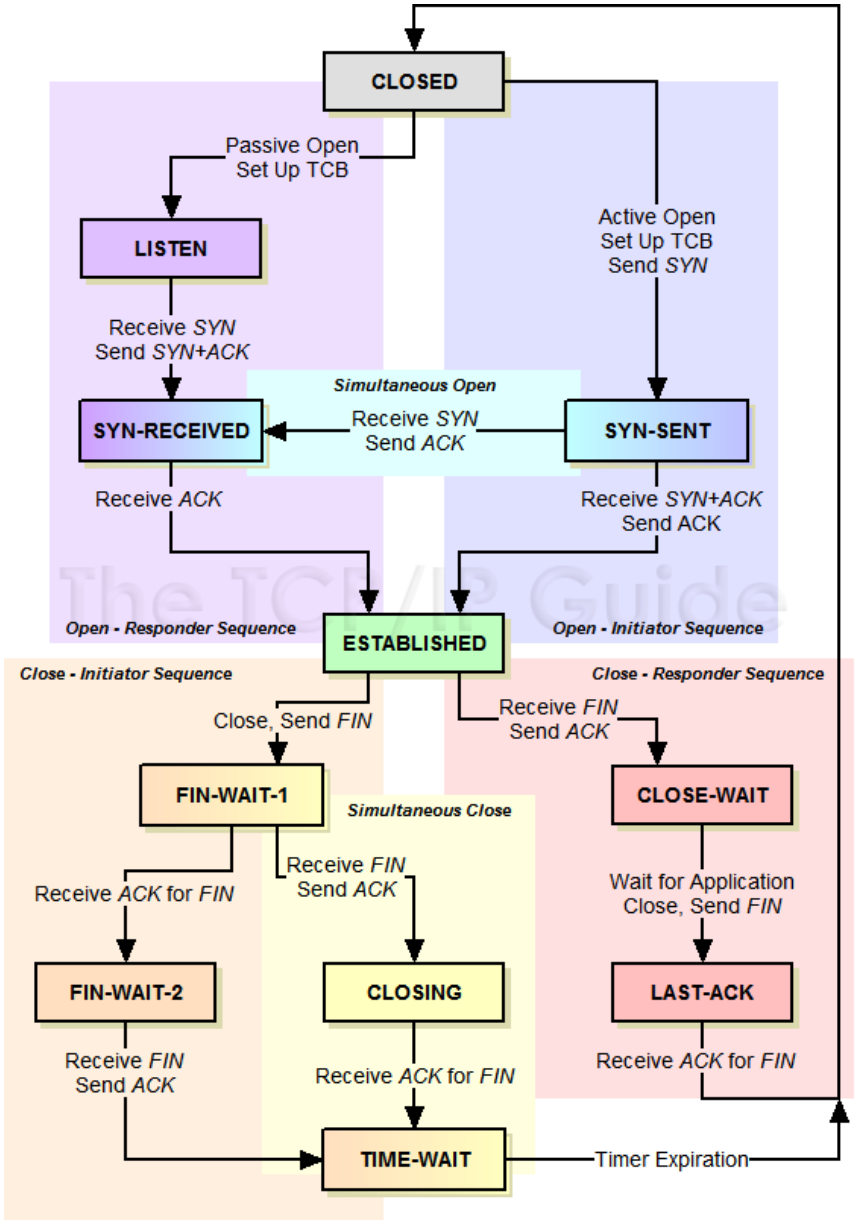
3. Connection Termination (Four-Way Handshake)

- **FIN Sent**: The client or server sends a FIN (finish) packet to indicate it has finished sending data.

- **ACK Received**: The receiver of the FIN packet sends an ACK packet to acknowledge the receipt of the FIN.
- **FIN Sent (Other End)**: The other end then sends its own FIN packet to indicate it has also finished sending data.
- **ACK Received**: The original sender of the first FIN packet sends an ACK packet to acknowledge the receipt of the second FIN, completing the termination process.

TCP FSM Diagram

Below is the FSM diagram representing TCP connection management. Each state and transition in the diagram corresponds to the actions described above.



Explanation of States

- **CLOSED**: Initial state, indicating no connection.
- **SYN_SENT**: Client has sent a SYN and is awaiting an ACK.
- **SYN_RCVD**: Server has received the SYN, sent an ACK, and is waiting for an ACK from the client.
- **ESTABLISHED**: Connection is established, and data can be exchanged.
- **FIN_WAIT_1**: Client has sent a FIN and is awaiting an ACK.
- **FIN_WAIT_2**: Client received an ACK for the first FIN and is awaiting a FIN from the server.
- **CLOSE_WAIT**: Server has received a FIN and is waiting to send an ACK.
- **LAST_ACK**: Server has sent its FIN and is waiting for an ACK from the client.
- **CLOSING**: Both sides have sent FIN packets, and each is awaiting the other's ACK.
- **TIME_WAIT**: Client waits to ensure the server received its ACK for the server's FIN.
- **CLOSED**: Final state after the connection is fully terminated.

9.a. Explain client/server and p-p architecture with a neat labeled diagram.

Client/Server Architecture

In the client/server architecture, there is a clear distinction between clients and servers. Clients request services, and servers provide these services. The server hosts, manages, and delivers resources, while clients access and use these resources.

Key Characteristics:

- **Centralized Control**: The server centralizes control and management.
- **Scalability**: Servers can handle multiple clients simultaneously.
- **Security**: Centralized control allows for better security management.
- **Reliability**: Servers can provide reliable and consistent services

Peer-to-Peer (P2P) Architecture

In a peer-to-peer (P2P) architecture, there is no central server. Instead, each node, referred to as a peer, acts as both a client and a server. Peers share resources directly with each other, enabling a decentralized network.

Key Characteristics:

- **Decentralized**: No central server; all peers have equal roles.
- **Scalability**: The network can grow by simply adding more peers.
- **Fault Tolerance**: If one peer fails, the network can still function.
- **Resource Sharing**: Resources are distributed across multiple peers.

Comparison

- **Client/Server**:
  - **Control**: Centralized.
  - **Management**: Easier to manage and secure.
  - **Performance**: Server can become a bottleneck.
  - **Reliability**: Single point of failure.
- **Peer-to-Peer**:
  - **Control**: Decentralized.
  - **Management**: Harder to manage and secure.
  - **Performance**: More scalable, as load is distributed.
  - **Reliability**: More fault-tolerant, as there is no single point of failure.

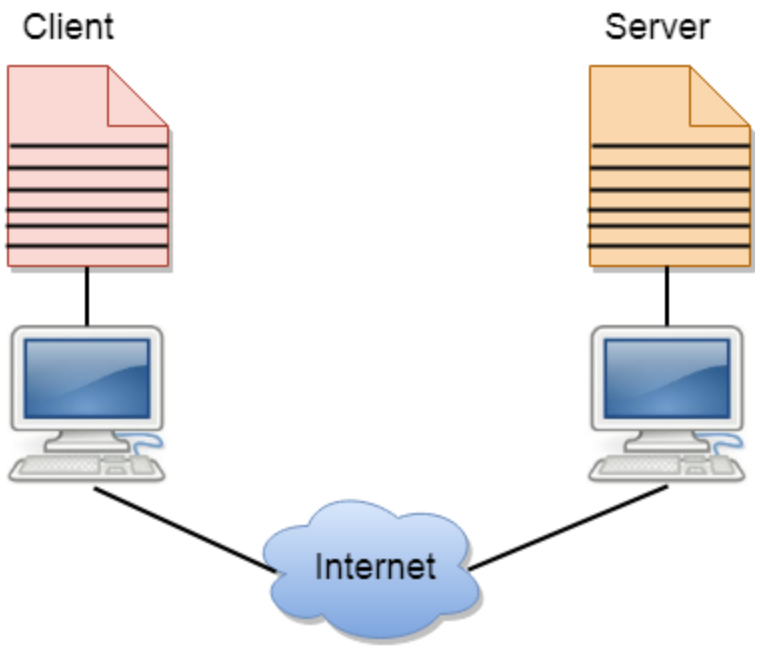
b. Explain use and server interaction with a neat diagram.

Client/Server Architecture

In a client/server architecture, clients request services and resources from a centralized server. The server processes these requests and responds to the clients. Here's a detailed example of how a web browser (client) interacts with a web server.

Example: Web Browser and Web Server

1. **Client (Web Browser)**:
   - Sends a request to the web server for a web page.
   - Receives and displays the web page.
2. **Server (Web Server)**:
   - Receives the client's request.
   - Processes the request (retrieves the requested web page).
   - Sends the requested web page back to the client.



Peer-to-Peer (P2P) Architecture

In a peer-to-peer architecture, all nodes (peers) are equal and can act as both clients and servers. Each peer can request resources from other peers and also provide resources to them.

Example: File Sharing Network

1. **Peer (A)**:
    - Requests a file from another peer (B).
    - Receives the file and can share it with others.
2. **Peer (B)**:
    - Receives the file request from peer (A).
    - Sends the requested file to peer (A).

10.a. Explain persistent and non persistent http in details.

Persistent and Non-Persistent HTTP

HTTP (Hypertext Transfer Protocol) is the foundation of data communication on the web. It defines how messages are formatted and transmitted, and how web servers and browsers should respond to various commands. Understanding the difference between persistent and non-persistent HTTP is crucial for optimizing web performance and resource usage.

Non-Persistent HTTP

In non-persistent HTTP, a separate TCP connection is established for each HTTP request/response pair. This means that for every resource (HTML page, image, script, etc.) requested from a web server, a new connection is opened, used, and then closed.

Characteristics:

1. **Multiple Connections**: Each request/response pair requires a new TCP connection.
2. **Connection Overhead**: Establishing and closing connections for each request adds significant overhead.
3. **Sequential Processing**: The client waits for a response to each request before making the next request.
4. **Simple Implementation**: Easier to implement and understand but less efficient.

Example:

When a web browser requests a web page containing multiple resources, such as images and scripts, it will open and close a TCP connection for each resource.

Persistent HTTP

In persistent HTTP (also known as HTTP keep-alive), a single TCP connection is used to send and receive multiple HTTP requests and responses. This connection remains open until it is explicitly closed by the client or server.

Characteristics:

1. **Single Connection**: Multiple requests and responses use the same TCP connection.
2. **Reduced Overhead**: Eliminates the need to repeatedly open and close connections.
3. **Parallel Processing**: Allows multiple requests to be sent in quick succession without waiting for responses.
4. **Improved Performance**: Reduces latency and improves overall performance.

Example:

When a web browser requests a web page, it can use the same TCP connection to request multiple resources from the server.

b. Write notes on:

(i) Email in the internet

(ii) Distributed DNS Architecture

(i) <u>Email in the Internet</u>

Email (electronic mail) is one of the most widely used communication tools on the internet. It allows users to send and receive messages through electronic devices. Here's a detailed explanation of how email works, the protocols involved, and a diagram illustrating the process.

How Email Works

1. **Composing**: The user writes an email using an email client (e.g., Gmail, Outlook).
2. **Sending**: The email client sends the message to an SMTP (Simple Mail Transfer Protocol) server.
3. **Transmitting**: The SMTP server processes the email and forwards it to the recipient's mail server.
4. **Receiving**: The recipient's mail server stores the email.
5. **Retrieving**: The recipient uses an email client to retrieve the email from their mail server using POP3 (Post Office Protocol) or IMAP (Internet Message Access Protocol).

Email Protocols

1. **SMTP (Simple Mail Transfer Protocol)**:
    - Used for sending emails.
    - Operates over port 25, 465 (SMTP over SSL), or 587 (SMTP over TLS).
2. **POP3 (Post Office Protocol version 3)**:
    - Used for retrieving emails.
    - Downloads emails from the server to the client and often deletes them from the server.
    - Operates over port 110 or 995 (POP3 over SSL).
3. **IMAP (Internet Message Access Protocol)**:
    - Used for retrieving emails.
    - Allows multiple clients to manage and synchronize mailboxes.
    - Operates over port 143 or 993 (IMAP over SSL).

Email Components

1. **Email Client**: Software or web application used to send and receive emails (e.g., Gmail, Outlook).
2. **Mail Server**: Server that processes and stores emails.
3. **Mail Transfer Agent (MTA)**: Software that transfers emails between servers.
4. **Mail User Agent (MUA)**: The email client used by the end user.
5. **DNS (Domain Name System)**: Translates domain names into IP addresses to locate mail servers.

(ii) Distributed DNS Architecture

The Domain Name System (DNS) is a hierarchical and decentralized naming system used to resolve human-readable domain names (like www.example.com) into machine-readable IP addresses (like 192.0.2.1). This resolution process is essential for the functioning of the internet.

Key Components of DNS

1. **DNS Resolver**: The client-side component that initiates the DNS query.
2. **Recursive DNS Server**: Intermediary that handles queries from DNS resolvers and performs the necessary lookups.

3. **Root DNS Server**: The top-level DNS server that responds to requests for the root domain (".").
4. **Top-Level Domain (TLD) DNS Server**: Manages domains for top-level domains like ".com", ".org", etc.
5. **Authoritative DNS Server**: Holds the DNS records for specific domains and provides the final answer to DNS queries.

How Distributed DNS Works

1. **DNS Query Initiation**:
    - A user types a URL (e.g., www.example.com) into a web browser.
    - The DNS resolver on the user's device sends a query to a recursive DNS server.
2. **Recursive DNS Server**:
    - The recursive DNS server checks its cache for the requested domain name.
    - If not found, it forwards the query to one of the root DNS servers.
3. **Root DNS Server**:
    - The root DNS server responds with a referral to a TLD DNS server responsible for the TLD (e.g., ".com").
4. **TLD DNS Server**:
    - The TLD DNS server responds with a referral to the authoritative DNS server for the specific domain (e.g., example.com).
5. **Authoritative DNS Server**:
    - The authoritative DNS server responds with the IP address associated with the domain name.
6. **DNS Resolution**:
    - The recursive DNS server caches the response and returns the IP address to the DNS resolver on the user's device.
    - The user's device uses the IP address to connect to the web server.