CBCS SCHEME

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

BCS502

Fifth Semester B.E./B.Tech. Degree Examination, Dec.2024/Jan.2025
Computer Networks

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FIVE full questions, choosing ONE full question from each module.
2. M : Marks , L: Bloom's level , C: Course outcomes.

		Module – 1	M	L	C
Q.1	a.	What is data communication? List and explain characteristics and components of communication model.	06	L1	CO1
	b.	Define switching. Explain Circuit Switched Network and Packet Switched Network.	06	L2	CO1
	c.	With neat sketch, explain different layers of TCP/IP protocol suite.	08	L2	CO1
		OR			
Q.2	a.	What are guided transmission media? Explain twisted pair cable in detail.	06	L1	CO1
	b.	What is Virtual Circuit Network (VCN)? With neat diagram, explain three phases involved in VCN.	08	L1	CO1
	c.	Write a note on Encapsulation and decapsulation at Source Host for TCP/IP protocol suite.	06	L2	CO1
		Module – 2			
Q.3	a.	Define Redundancy. Explain CRC encoder and CRC decoder operation with block diagram.	08	L2	CO2
	b.	Distinguish between Flow Control and Error Control. Explain Stop and Wait Protocol.	08	L2	CO2
	c.	List and explain Control Fields of I-frames, S-frames and U-frames.	04	L2	CO2
		OR			
Q.4	a.	What is Hamming distance? With example, explain Parity Check Code.	06	L1	CO2
	b.	Define Framing. Explain character oriented framing and bit-oriented framing.	06	L1	CO2
	c.	With flow diagram, explain CSMA/CA.	08	L2	CO2
		Module – 3			
Q.5	a.	Explain virtual-circuit approach to route the packets in packet-switched network.	10	L2	CO3
	b.	Illustrate the working of OSPF and BGP.	10	L3	CO3
		OR			
Q.6	a.	Explain IPv6 datagram format.	10	L2	CO3
	b.	Write an Dijikstra's algorithm to compute shortest path through graph.	06	L1	CO3
	c.	Write a note on Routing Information Protocol (RIP) algorithm.	04	L2	CO3
		Module – 4			
Q.7	a.	Explain Go-Back-N protocol working.	10	L2	CO4
	b.	With neat sketch, explain three-way handshaking of TCP connection establishment.	10	L2	CO4

		OR			
Q.8	**a.**	With an outline, explain selective repeat protocol.	**10**	**L2**	**CO4**
	b.	List and explain various services provided by User Datagram Protocol (UDP).	**10**	**L2**	**CO4**
		Module – 5			
Q.9	**a.**	Briefly explain Secure Shell (SSH).	**10**	**L2**	**CO4**
	b.	Write a note on Request message and response message formats of HTTP.	**10**	**L2**	**CO4**
		OR			
Q.10	**a.**	With neat diagram, explain the basic model of FTP.	**04**	**L2**	**CO4**
	b.	Describe the architecture of electronic mail (e-mail).	**06**	**L3**	**CO4**
	c.	Briefly explain Recursive Resolution and Iterative Resolution in DNS.	**10**	**L2**	**CO4**

* * * * *

Computer Network (BCS-502)

Scheme & Solution

Session:-Dec24/Jan25

1a) What is data communication? List and explain characteristics and components of data communication. (6Marks)

Defination-1 Marks

Characteristics & Component-3 Marks

Diagram-1 Marks

Example-1 Marks

1a) Data communication is the process of transmitting information electronically between two or more points. This information can be in the form of text, images, audio, video, or any other digital format.

Characteristics of data communication include:

•**Accuracy**: The data received should be the same as the data sent, without errors or distortion.

•**Timeliness**: The data should be delivered within an acceptable timeframe.

•**Efficiency**: The transmission process should be optimized for bandwidth usage and minimize overhead.

•**Security**: The data should be protected from unauthorized access and tampering.

•**Reliability**: The system should be robust and able to handle failures or interruptions.

Components of a data communication system typically consist of:

•**Sender**: The device or entity that originates the data transmission.

•**Receiver**: The device or entity that receives the transmitted data.

•**Medium**: The physical pathway or channel through which the data travels. This can include copper wires, fiber optic cables, radio waves, or satellite links.

•**Protocol**: A set of rules and conventions that govern the format and procedures for data exchange. Protocols ensure interoperability between different devices and systems.

•**Message**: The information being transmitted, which can be in various forms, such as text, images, audio, or video.

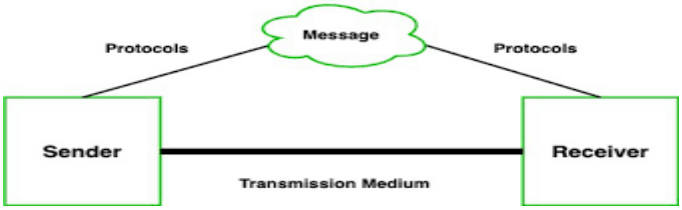


Figure: - Components of Data Communication

1b) What is networking? Explain Circuit Switched Network and Packet Switched Network with neat sketch. (6 Marks)

Defination-1 Marks

Explaintaion-3 Marks

Diagram-2 Marks

1b) Networking is the interconnection of two or more computing devices to share resources, such as printers, data files, and applications. This interconnection can be established using wired or wireless media. The core purpose of networking is to facilitate communication and data exchange between devices.

Circuit-Switched Networks: A circuit-switched network establishes a dedicated connection between two devices before any data transmission occurs. This connection reserves resources (bandwidth and processing capacity) along the entire path, ensuring a constant and predictable data flow.2 This dedicated pathway remains active for the entire duration of the communication session, similar to a dedicated phone line. Once the session ends, the circuit is torn down and the resources are released.

Circuit switching is suitable for applications that require real-time, uninterrupted communication, like traditional phone calls. However, it can be inefficient if the connection is idle for long periods, as the allocated resources remain unused.

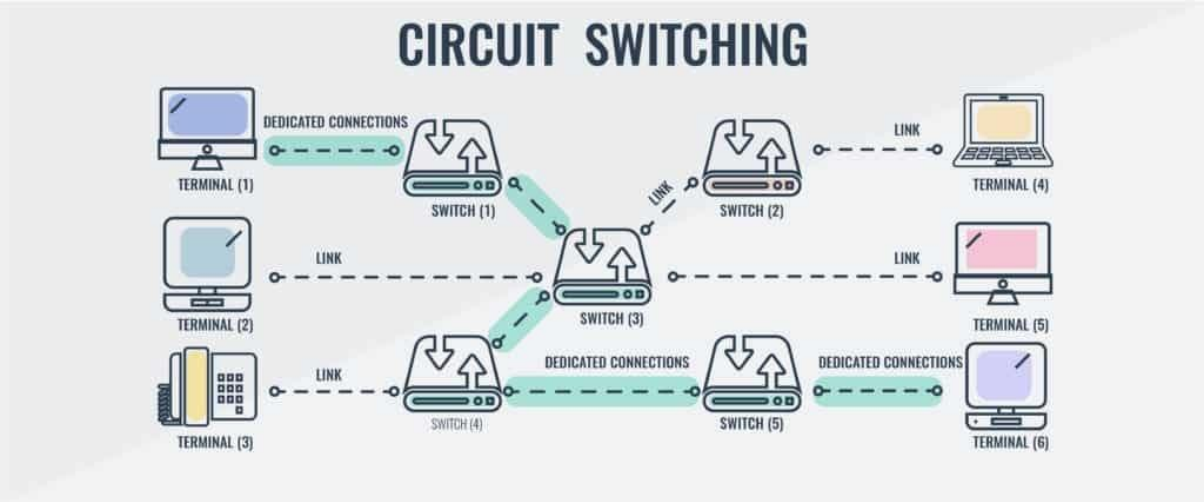


Figure: - Circuit Switching Network

Packet-Switched Networks: A packet-switched network divides data into small units called packets, each containing a portion of the data along with addressing information. These packets are then transmitted independently over the network and can travel different paths to reach their

destination.2 At the destination, the packets are reassembled in the correct order to reconstruct the original data.

Packet switching allows for more efficient use of network resources as multiple devices can share the same physical links. It is also more adaptable to network congestion as packets can be rerouted dynamically. However, packet switching can introduce latency as packets may arrive out of order or experience delays. The Internet is a prime example of a packet-switched network.

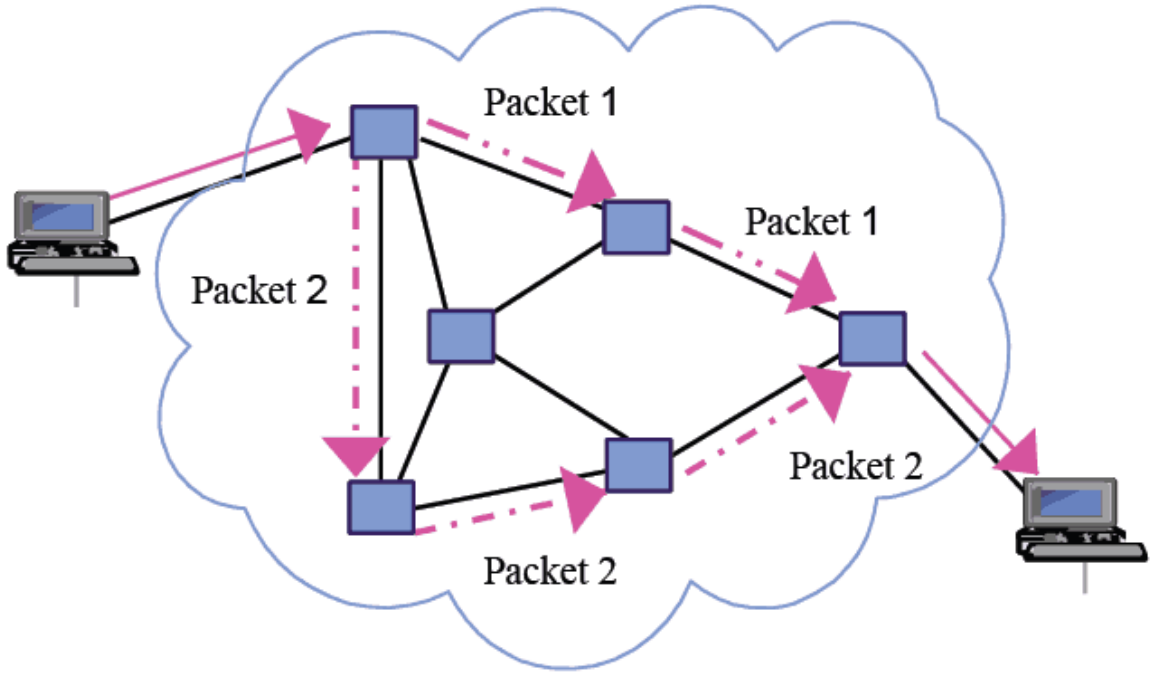


Figure: - Packet Switching Network

1C) With neat sketch, explain different layer of TCP/IP protocol suite. (8 Marks)

Layer-2 Marks

Explaination-4 Marks

Diagram-2 Marks

1C) The TCP/IP protocol suite is a hierarchical model that describes the rules and conventions for communication over a network. It consists of five layers, each responsible for specific functions. Here's a sketch illustrating the different layers of the TCP/IP protocol suite, followed by an explanation of each layer:

1. Physical Layer:

•This layer deals with the physical transmission of data over the communication medium.

•It defines the electrical, mechanical, and procedural aspects of the physical link, including cable types, connectors, voltage levels, and signalling methods.

•Examples of technologies operating at this layer include Ethernet cables, fiber optic cables, and wireless radio waves.

•The physical layer transmits individual bits as electrical or optical signals.

•Switching at the physical layer is circuit switching.

2. Data Link Layer:

•This layer provides reliable transmission of data frames between adjacent nodes on a network link.

•It is responsible for framing, addressing (using MAC addresses), error detection and correction, and flow control at the link level.

•The data link layer is divided into two sublayers: **Logical Link Control (LLC)** and **Media Access Control (MAC)**.

◦**LLC** handles flow and error control.

◦**MAC** manages access to the shared medium in broadcast networks.

•Packet switching at the data link layer is typically done using a virtual-circuit approach.

3. Network Layer:

•This layer handles the routing of data packets across multiple networks.

•It is responsible for addressing (using IP addresses), routing (determining the path for packets), and fragmentation and reassembly of packets.

•The primary protocol at this layer is the **Internet Protocol (IP)**.

•**IP** is a connectionless protocol that does not provide flow control, error control, or congestion control.

•The network layer uses packet switching, typically using the datagram approach.

4. Transport Layer:

•This layer provides end-to-end communication between processes running on different hosts.

•It is responsible for segmentation and reassembly of data, flow control, error control, and congestion control at the end-to-end level.

•Two main transport layer protocols are **TCP (Transmission Control Protocol)** and **UDP (User Datagram Protocol)**.

◦**TCP** is a reliable, connection-oriented protocol that provides flow control, error control, and congestion control.

◦**UDP** is an unreliable, connectionless protocol that provides minimal overhead and is suitable for applications where reliability is less critical.

5. Application Layer:

•This layer provides services to end users and applications.

•It includes protocols for various applications, such as email (SMTP, POP3), file transfer (FTP), web browsing (HTTP), and domain name resolution (DNS).

•Application layer protocols are built on top of the transport layer and can use either TCP or UDP depending on their requirements.

These layers work together to enable communication over networks, ensuring that data is transmitted reliably, efficiently, and securely.

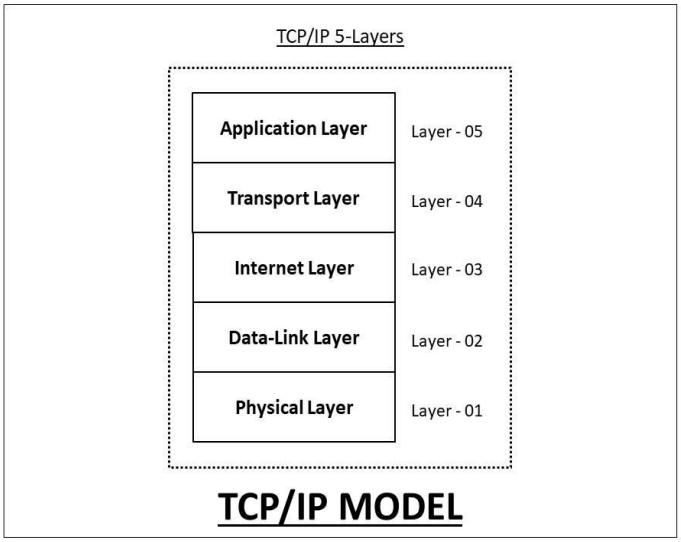


Figure: - TCP/IP Protocol Suite

2a) What are guided transmission media? Explain twisted pair cable in detail. (6 Marks)

Defination-2 Marks

Explaintaion-4Marks

2a) Guided transmission media are the physical means through which data is transmitted from one device to another. These media provide a pathway that guides the data signals and are often used in wired communication. Examples include twisted pair cables, coaxial cables, and optical fiber cables.

Twisted Pair Cable

A **twisted pair cable** is one of the most common types of guided transmission media, widely used in telecommunication and networking. It consists of pairs of insulated copper wires twisted together. The twisting reduces electromagnetic interference (EMI) from external sources and crosstalk between adjacent pairs.

Structure of a Twisted Pair Cable:

1. **Copper Wires**: The core conductors are typically copper, which carries the electrical signals.
2. **Insulation**: Each wire is coated with an insulating material to prevent short circuits.
3. **Twisting**: The two wires are twisted together to minimize interference.

4. **Sheath**: An outer protective covering surrounds the twisted pairs for protection against physical damage.

Types of Twisted Pair Cable:

1. **Unshielded Twisted Pair (UTP)**:
    - No additional shielding.
    - Cheaper and easier to install.
    - More susceptible to interference.
    - Commonly used in Ethernet networks and telephone lines.
2. **Shielded Twisted Pair (STP)**:
    - Includes an additional metallic shield around the twisted wires to reduce EMI.
    - Provides better protection from interference but is more expensive.
    - Used in environments with high interference.

Advantages of Twisted Pair Cable:

- **Cost-Effective**: UTP cables are inexpensive and widely available.
- **Easy to Install**: Lightweight and flexible, making them easy to handle.
- **Suitable for Short Distances**: Performs well in short to medium-length transmissions.

Disadvantages:

- **Limited Bandwidth**: Cannot support very high data rates over long distances.
- **Susceptible to Noise**: UTP cables are more prone to interference compared to other guided media like coaxial cables or optical fibres.
- **Distance Limitation**: Signal attenuation occurs over long distances, requiring repeaters or amplification.

Applications of Twisted Pair Cable:

- **Local Area Networks (LANs)**: Common in Ethernet networking.
- **Telephone Lines**: Used in traditional telephone systems.
- **DSL Connections**: Twisted pairs are used in DSL broadband technology.

2b) What is Virtual Circuit Network (VCN)? With neat diagram explain three phases involved in VCN. (8 Marks)

Defination-2 Marks

Explaination-4 Marks

Diagram-2 Marks

2b) A **Virtual Circuit Network (VCN)** is a type of packet-switched network where a logical connection is established between the sender and receiver before any data transmission begins. This logical connection behaves like a dedicated physical circuit, but it is virtual, meaning the data packets still share the physical medium with other communications.

VCNs are commonly used in technologies like ATM (Asynchronous Transfer Mode), Frame Relay, and MPLS (Multiprotocol Label Switching).

Key Features of Virtual Circuit Network

1. **Connection-Oriented Service**: A connection is established before data transmission.
2. **Fixed Path**: All packets follow the same path during the session, ensuring they arrive in order.
3. **State Maintenance**: Routers or switches maintain the state of each virtual circuit.
4. **Efficiency**: Ensures a logical connection, making it suitable for real-time communication.

Three Phases of Virtual Circuit Network

1. **Call Setup Phase**
    - A virtual connection is established between the source and the destination.
    - A unique **Virtual Circuit Identifier (VCI)** is assigned to identify the connection.
    - Routing tables in intermediate switches/routers are updated to record the path.
    - This phase ensures that resources such as bandwidth and buffers are reserved.

2.Data Transfer Phase

- Once the connection is established, data packets are sent from the source to the destination.
- Each packet contains the VCI, allowing switches/routers to forward packets along the predetermined path.
- Packets are delivered in order since the path is fixed

3.Call Termination Phase

- After the data transfer is complete, the virtual connection is terminated.
- The resources reserved during the call setup phase (e.g., bandwidth) are released.
- The routing tables in intermediate devices are updated to remove the virtual circuit.

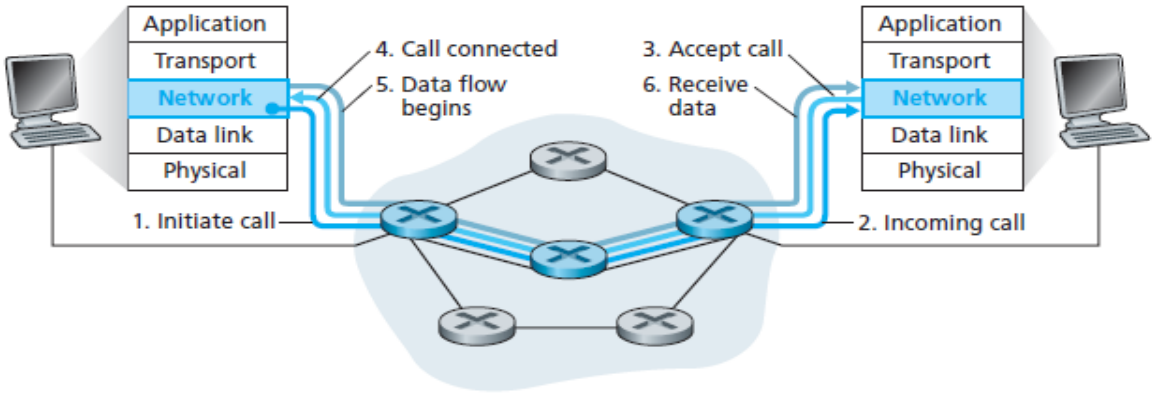


Figure: - Phases involved in VCN

2c) Write a note on encapsulation and decapsulation at a source host at TCP/IP protocol suite. (6 Marks)

Explaination-4 Marks

Diagram-2 Marks

2c) **Encapsulation and Decapsulation in the TCP/IP Protocol Suite**

The TCP/IP protocol suite is a layered model for networking communication. The process of **encapsulation** and **decapsulation** ensures that data is properly prepared for transmission and received correctly at the destination.

Encapsulation

Encapsulation is the process of adding headers (and sometimes footers) to data as it moves through the layers of the TCP/IP protocol stack at the source host. Each layer performs specific functions and appends relevant information to ensure successful data transmission.

Steps in Encapsulation:

1. **Application Layer**:
    - Data is generated at the application layer (e.g., HTTP, FTP).
    - No specific header is added here in the TCP/IP model.
2. **Transport Layer**:
    - The transport layer (e.g., TCP or UDP) divides the data into smaller segments.
    - A **TCP header** or **UDP header** is added, containing information like port numbers, sequence numbers, and checksums for error detection.
    - The encapsulated unit at this layer is called a **segment**.
3. **Network Layer**:
    - The network layer (IP) adds an **IP header**, which contains the source and destination IP addresses, among other routing information.

- The encapsulated unit at this layer is called a **packet**.

4. **Data Link Layer**:
    - The data link layer (e.g., Ethernet, Wi-Fi) adds a **frame header** and a **frame trailer**.
    - The frame header contains source and destination MAC addresses, while the trailer often contains a cyclic redundancy check (CRC) for error checking.
    - The encapsulated unit at this layer is called a **frame**.
5. **Physical Layer**:
    - The data link layer frame is converted into electrical, optical, or radio signals for physical transmission.

Decapsulation

Decapsulation is the reverse process of encapsulation, performed at the destination host. Each layer removes its respective header/trailer and processes the data before passing it to the next layer.

Steps in Decapsulation:

1. **Physical Layer**:
    - The physical layer receives the signals and converts them back into bits, which are passed to the data link layer.
2. **Data Link Layer**:
    - The data link layer removes the **frame header** and **trailer**, checks for errors, and passes the remaining **packet** to the network layer.
3. **Network Layer**:
    - The network layer removes the **IP header**, checks the destination IP address, and passes the remaining **segment** to the transport layer.
4. **Transport Layer**:
    - The transport layer removes the **TCP/UDP header**, reassembles the data if it was segmented, and passes it to the application layer.
5. **Application Layer**:
    - The application layer processes the data for use by the end-user application.

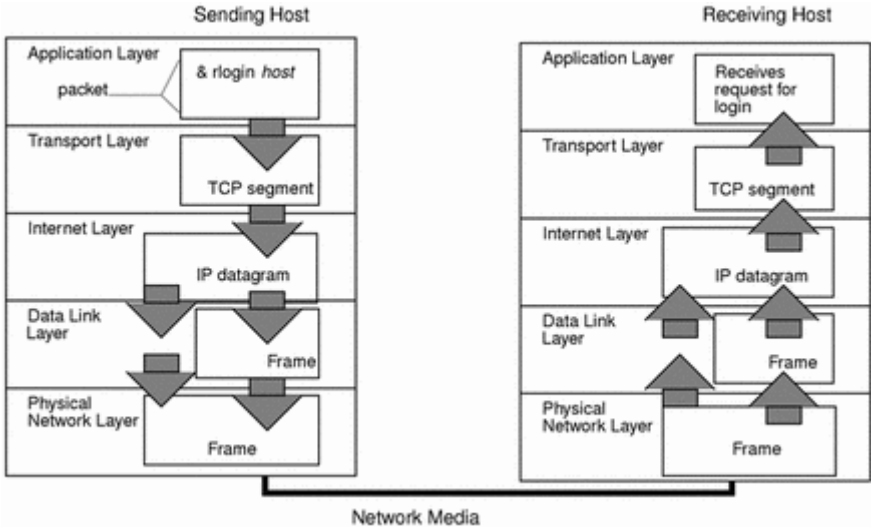


Figure: -Encapsulation & Dencapsulation

3a) Define Redundancy. Explain CRC encoder and CRC decoder operation with block diagram. (8 Marks)

Defination-1 Marks

Explaination-5 Marks

Diagram-1 Marks

Example-1 Marks

3a) Redundancy: - In networking and error detection, redundancy refers to the inclusion of extra bits or information in transmitted data to detect and possibly correct errors. Redundant bits do not carry new information but help the receiver verify the accuracy of the received data.

Cyclic Redundancy Check (CRC)

CRC is an error-detecting code that adds a sequence of redundant bits (called a **CRC code** or **check bits**) to the transmitted data. These bits are generated using polynomial division and allow the receiver to detect errors in the received data

One common method of error detection using redundancy is Cyclic Redundancy Check (CRC), a powerful technique for detecting errors in transmitted data.

CRC Encoder

The **CRC Encoder** adds the redundant bits to the original data using the following process:

1. **Input Data (Message)**:
    - The binary message to be transmitted is represented as a polynomial (e.g., D(x)D(x)D(x)).
2. **Generator Polynomial**:
    - A predefined binary divisor, represented as a polynomial G(x)G(x)G(x), is agreed upon by both sender and receiver.

3. **Appending Zeros**:
    - The message is appended with n−1n-1n−1 zeros, where nnn is the degree of G(x)G(x)G(x).
4. **Division**:
    - The message (with appended zeros) is divided by G(x)G(x)G(x) using **modulo-2 division** to obtain the remainder.
5. **Transmitted Frame**:
    - The remainder (CRC bits) is appended to the original message to form the **transmitted frame**.

CRC Decoder

The **CRC Decoder** at the receiver side performs the following operations to verify the integrity of the received data:

1. **Input Frame**:
    - The received frame consists of the original message and the appended CRC bits.
2. **Division**:
    - The received frame is divided by the same generator polynomial G(x)G(x)G(x) using **modulo-2 division**.
3. **Error Detection**:
    - If the remainder of the division is **zero**, the data is considered error-free.
    - If the remainder is **non-zero**, it indicates an error in the transmission.

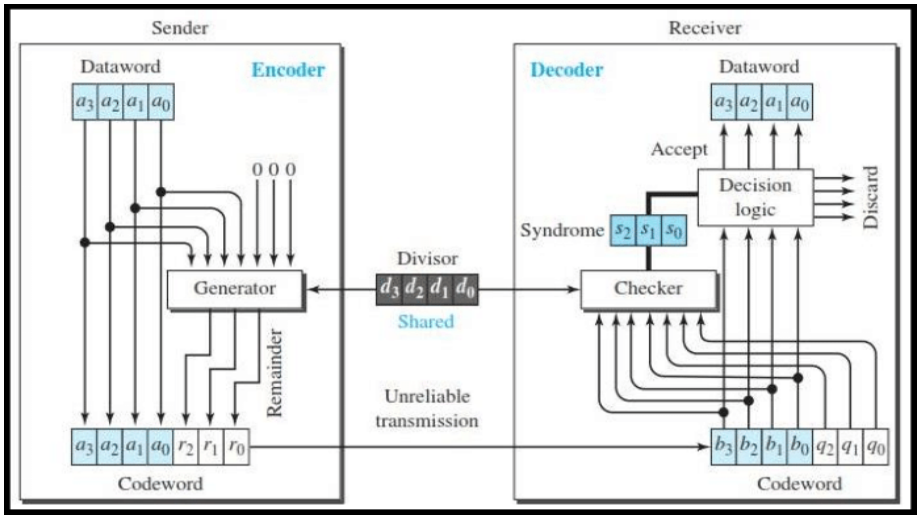


Figure: - CRC Encoder & Decoder

3b) Distinguish between Flow Control & Error control. Explain Stop & Wait protocol. (8 Marks)

Differences-1 *8=8Marks

3b)

Aspect	Flow Control	Error Control
Definition	Flow control manages the pace of data transmission to ensure the sender does not overwhelm the receiver.	Error control ensures the accurate delivery of data by detecting and correcting errors in the transmitted frames.
Focus	Prevents buffer overflow or underflow at the receiver.	Detects and retransmits corrupted or lost frames.
Primary Goal	Maintains synchronization between sender and receiver for smooth data transfer.	Ensures reliable delivery of error-free data.
Methods	Examples: Stop-and-Wait, Sliding Window (for flow control).	Examples: Automatic Repeat Request (ARQ), Forward Error Correction (FEC).
Scope	Deals with the timing and rate of data transfer.	Deals with data integrity and retransmission.

3c) List & explain control fields of I-Frames-Frames & U-Frames. (4 Marks)

Explanation-3 Marks

Diagram-3 Marks

3c) The control field in data link layer frames is used to specify the type of frame and manage communication between devices. In the **High-Level Data Link Control (HDLC)** and similar protocols, there are three main types of frames:

1. **I-Frames (Information Frames)**: Used to transmit user data and control information.
2. **S-Frames (Supervisory Frames)**: Used for flow and error control.
3. **U-Frames (Unnumbered Frames)**: Used for control and management, such as establishing or disconnecting a connection.

Control Field Format for I-Frames

I-frames are primarily used for data transfer. Their control field contains the following:

Bit(s)	Field	Description
1	**Type Identifier** (0)	The first bit is always 0, indicating an I-frame.
N(S)	**Send Sequence Number**	Specifies the sequence number of the frame being sent.
P/F	**Poll/Final Bit**	Used for polling (by sender) or final acknowledgment (by receiver).

Bit(s)	Field	Description
N(R)	**Receive Sequence Number**	Acknowledges the last successfully received frame.

Key Features:

- **N(S)** ensures proper sequencing of frames sent.
- **N(R)** enables acknowledgment and retransmission of lost or corrupted frames.

Control Field Format for S-Frames

S-frames are used for supervisory functions like acknowledgment and flow control. Their control field is structured as follows:

Bit(s)	Field	Description
2	**Type Identifier** (10)	The first two bits indicate an S-frame.
2	**Code**	Specifies the function (RR, RNR, REJ, SREJ).
P/F	**Poll/Final Bit**	Used for polling or signalling the end of a response.
N(R)	**Receive Sequence Number**	Acknowledges the last received frame.

Supervisory Functions:

1. **RR (Receive Ready)**: Indicates the receiver is ready to receive more frames.
2. **RNR (Receive Not Ready)**: Indicates the receiver is not ready to receive frames due to a buffer full or another condition.
3. **REJ (Reject)**: Requests retransmission starting from a specific frame (negative acknowledgment).
4. **SREJ (Selective Reject)**: Requests retransmission of a single specific frame.

Control Field Format for U-Frames

U-frames are used for managing control tasks like establishing or terminating connections. Their control field is flexible and varies based on the operation.

Bit(s)	Field	Description
2	**Type Identifier** (11)	The first two bits indicate a U-frame.
Varying	**Control Command/Response**	Specifies the type of control function (e.g., SABME, DISC).
P/F	**Poll/Final Bit**	Used for polling or signalling the end of a response.

Common U-Frame Commands and Responses:

1. **SABME (Set Asynchronous Balanced Mode Extended)**: Establishes a logical connection.
2. **DISC (Disconnect)**: Terminates a logical connection.
3. **UA (Unnumbered Acknowledgment)**: Acknowledges a command.
4. **FRMR (Frame Reject)**: Indicates an invalid frame.
5. **DM (Disconnected Mode)**: Indicates the device is not ready.

4a) What is a Hamming distance? With example, explain Parity Check Code. (6 Marks)

Defination-1 Marks

Explanation-3 Marks

Diagram-1 Marks

Example-1 Marks

4a) The **Hamming Distance** is a measure of the difference between two binary strings of equal length. It is defined as the number of bit positions where the corresponding bits are different. It is widely used in error detection and correction techniques.

Mathematical Definition:

If two binary strings AAA and BBB have lengths n, the Hamming Distance d is given by:

d=Number of positions where A[i]≠B[i].

Binary Strings:

- A=10101A = 10101A=10101
- B=11100B = 11100B=11100

Hamming Distance d:

- d=2d = 2d=2 (two bits differ).

Parity Check Code

Parity Check Code is a simple error detection method that adds a **parity bit** to a set of data bits to ensure that the total number of 1's (or 0's) in the codeword is even or odd. It detects single-bit errors effectively but cannot correct them.

Types of Parity:

1. **Even Parity**:
    - The parity bit is chosen such that the total number of 1's in the codeword is even.
2. **Odd Parity**:
    - The parity bit is chosen such that the total number of 1's in the codeword is odd.

Parity Check Code Operation:

1. **Sender Side (Encoding)**:
    - The sender calculates the parity bit based on the data bits.
    - The parity bit is appended to the original data to form a **codeword**.
2. **Receiver Side (Decoding)**:
    - The receiver checks the received codeword for parity.
    - If the parity condition is violated, an error is detected.

Example of Even Parity Check Code:

1. **Original Data**:
    - D=110101D = 110101D=110101 (6 bits).
2. **Calculate Parity Bit**:
    - Count the number of 1's: 110101110101110101 → 4 ones (even).
    - For **even parity**, no additional 1 is needed. Add a parity bit of 000.
3. **Codeword**:
    - C=1101010C = 1101010C=1101010.
4. **Receiver Side**:
    - If the received codeword is 110101011010101101010:
        - Count the number of 1's: 4 (even). No error detected.
    - If the received codeword is 110100011010001101000:
        - Count the number of 1's: 3 (odd). Error detected.

Example of Odd Parity Check Code:

1. **Original Data**:
    - D=101010D = 101010D=101010 (6 bits).
2. **Calculate Parity Bit**:
    - Count the number of 1's: 101010101010101010 → 3 ones (odd).
    - For **odd parity**, no additional 1 is needed. Add a parity bit of 000.
3. **Codeword**:
    - C=1010100C = 1010100C=1010100.
4. **Receiver Side**:
    - If the received codeword is 101010010101001010100:
        - Count the number of 1's: 3 (odd). No error detected.

- If the received codeword is 10101101010110101010110:
    - Count the number of 1's: 4 (even). Error detected.

4b) **Define framing. Explain character-oriented farming & bit-oriented framing. (6 Marks)**

Defination-1 Marks

Explanation-4 Marks

Example-1 Marks

4b) Framing: - Framing is the process in the **Data Link Layer** of dividing a continuous stream of bits into manageable data units called **frames**. These frames are then transmitted across the network, ensuring proper synchronization, error detection, and efficient data communication.

Types of Framing

There are two main types of framing mechanisms:

1. **Character-Oriented Framing**: Uses characters (typically ASCII) as control signals for marking the start and end of a frame.
2. **Bit-Oriented Framing**: Uses bit patterns to define and manage the boundaries of a frame.

Character-Oriented Framing

Character-Oriented Framing (also called byte-oriented framing) uses special characters from the character set (like ASCII) to indicate the beginning and end of a frame.

Key Concepts:

1. **Control Characters**:
    - Special characters like **STX (Start of Text)** and **ETX (End of Text)** mark the start and end of the frame, respectively.
2. **Data Representation**:
    - Data is transmitted as a sequence of characters (bytes).
3. **Transparency**:
    - If the data contains control characters (like STX or ETX), **byte stuffing** is used to differentiate them from actual frame delimiters.

Bit-Oriented Framing

Bit-Oriented Framing uses specific **bit patterns** to identify the start and end of a frame, regardless of the content inside the frame.

Key Concepts:

1. **Bit Pattern**:
    - A predefined bit sequence (e.g., **01111110**) called a **flag** is used to mark the start and end of the frame.

2. **Transparency**:
    - If the data contains the flag sequence, **bit stuffing** is used. This involves inserting an extra bit (0) after five consecutive 1's in the data stream to prevent confusion.
3. **Frame Format**:
    - The frame is bounded by flag sequences, with the data in between.

4c) With flow diagram, explain CSMA/CD. (8 Marks)

Defination-1 Marks

Explanation-5 Marks

Diagram-2 Marks

4c) **CSMA/CD** is a media access control protocol used in traditional Ethernet networks to manage data transmission on a shared communication channel. It ensures that devices avoid collisions when transmitting data and resolve them efficiently if they occur.

How CSMA/CD Works

The CSMA/CD algorithm operates in three main steps:

1. **Carrier Sense**: A device checks whether the channel is idle (no data is being transmitted) before attempting to send data.
2. **Multiple Access**: Multiple devices share the same channel.
3. **Collision Detection**: If two devices transmit simultaneously, a collision is detected, and both devices stop transmission and back off for a random period before retrying.

Flow Diagram of CSMA/CD

Below is a textual description of the flow, which can be visualized in the diagram:

1. **Start**: A device has data to send.
2. **Sense the Channel**:
    - If the channel is idle:
        - Begin transmission.
    - If the channel is busy:
        - Wait until it becomes idle.
3. **Transmit Data**:
    - Continuously monitor the channel during transmission.
4. **Collision Detection**:
    - If no collision is detected:

- Transmission is successful, and the process ends.

- If a collision is detected:
  - Send a **jam signal** to notify all devices of the collision.
  - Stop transmission.

5. **Backoff**:
   - Wait for a random backoff time (based on the binary exponential backoff algorithm).
   - Retry transmission after the backoff period.
6. **Repeat**: Return to sensing the channel and repeat the process until the data is transmitted successfully.

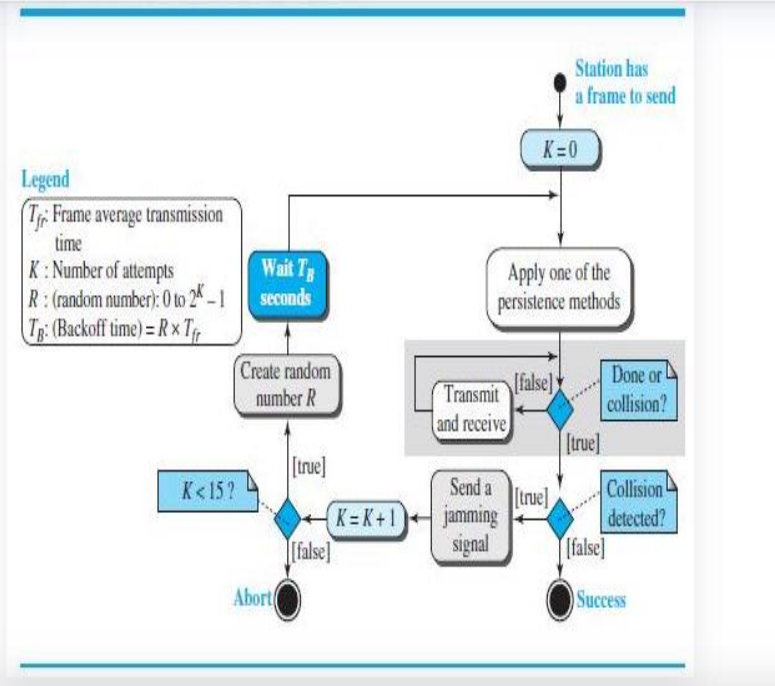


Figure: - Flow diagram for CSMA/CD

5a) Explain virtual circuit approach to route the packets in packet switching networks. (10 Marks)

Explanation-6 Marks

Diagram-4 Marks

Example-1 Marks

(VC) approach is one of the two main strategies used in packet-switched networks for routing packets (the other being **Datagram Switching**). In the Virtual Circuit approach, a logical path (virtual circuit) is established between the source and destination before any data packets are transmitted. This approach

is primarily used in connection-oriented networks such as **Frame Relay** and **ATM** (Asynchronous Transfer Mode).

Key Concepts of Virtual Circuit Approach

1. **Connection Setup**:
   - Before data transmission begins, a **virtual circuit** is established between the source and destination. This involves negotiating the path through the network.
   - The network devices (routers or switches) maintain a state table to store routing information for the duration of the communication.
2. **Packet Routing**:
   - Once the virtual circuit is established, all packets from the source to the destination follow the same pre-established path.
   - Each packet is assigned a unique **VCI (Virtual Circuit Identifier)** to differentiate it from other virtual circuits, even if the packets travel over the same physical links.
3. **Data Transfer**:
   - During data transfer, the packets carry this **VCI** instead of a complete destination address.
   - Routers or switches use the VCI to determine where to forward each packet.
4. **Connection Teardown**:
   - After data transmission is complete, the virtual circuit is **terminated**, and the resources allocated for the connection are freed.
   - The teardown process involves notifying all the intermediate devices and releasing any state information.

Steps in Virtual Circuit Packet Switching

1. **Connection Setup Phase**:
   - The source sends a **setup request** to the destination.
   - Intermediate routers/switches allocate resources (such as buffers and paths) and forward the request.
   - Once the destination acknowledges, the connection is established.
2. **Data Transfer Phase**:
   - The source sends data packets, each with a **VCI**.
   - Intermediate routers use the VCI to forward packets to the next hop along the established path.
   - Each router maintains a **routing table** for the virtual circuit that maps the VCI to the outgoing link.
3. **Connection Teardown Phase**:

- When the communication is complete, a **teardown request** is sent from the source or destination.
- The teardown message propagates back through the network to release resources and delete routing tables.

Advantages of Virtual Circuit Approach

1. **Guaranteed Path**:
   - Since a dedicated path is established, packets are guaranteed to follow the same route, avoiding the uncertainty of dynamic routing.
2. **Efficient Use of Resources**:
   - Network resources (like buffers and links) can be allocated and managed effectively since the path is predefined and consistent.
3. **Better Quality of Service (QoS)**:
   - Virtual circuits allow for better management of network traffic and quality of service, as each path can be prioritized.
4. **Reliable Data Delivery**:
   - The connection-oriented nature of virtual circuits provides reliable data delivery and may include error handling mechanisms.

Disadvantages of Virtual Circuit Approach

1. **Overhead in Setup**:
   - The need for an initial connection setup introduces overhead and delays, which can affect time-sensitive applications.
2. **Resource Consumption**:
   - Each virtual circuit requires a dedicated set of resources (routing tables, bandwidth), which can be inefficient for networks with large numbers of short-lived connections.
3. **Scalability Issues**:
   - The system may struggle to scale with a large number of active virtual circuits, as routers need to store and manage state information for each active connection.
4. **Fixed Path**:
   - Since the path is established ahead of time, if network conditions change (such as a link failure), the path may no longer be optimal or may require a new connection setup.

Example of Virtual Circuit Approach

1. **Connection Setup**:
   - Source A sends a setup request to the destination B.
   - Routers R1, R2, and R3 allocate paths and establish the virtual circuit.

2. **Data Transfer**:

- Packets from source A to destination B are transmitted with the VCI in the header. Each router uses the VCI to forward the packet along the established path.

3. **Connection Teardown**:

- After the data transfer, source A or destination B sends a teardown signal.
- Routers R1, R2, and R3 delete the VCI routing information and free resources.

5b) Illustrate the working of OSPF & BGP (10 Marks)

Explanation- 5+5 Marks

5b) **OSPF (Open Shortest Path First)** is a link-state routing protocol used to find the best path for data exchange in an IP network. It is commonly used in large enterprise networks because of its scalability and efficient operation.

OSPF Working Process

1. **Neighbor Discovery**:

- OSPF routers send **Hello packets** to discover and establish neighbor relationships with routers on the same network segment. The routers exchange information about their states and their neighbors.
- A router can only communicate with its direct neighbors before sharing routing information.

2. **Link-State Advertisement (LSA)**:

- After establishing a neighbor relationship, routers begin sharing their **Link-State Advertisements (LSA)**, which contain information about their links (interfaces) and network topology.
- Each router sends LSA to all of its neighbors, which then forward this information to their neighbors. This process helps in creating a complete map of the network.

3. **Building the Link-State Database**:

- Each router collects all LSAs from neighboring routers and builds a **Link-State Database (LSDB)**, which contains the network's topology.
- This database is identical across all routers in an OSPF area, ensuring that each router has the same view of the network.

4. **Shortest Path First (SPF) Calculation**:

- Once the LSDB is populated, OSPF uses the **Dijkstra algorithm** (also known as **SPF algorithm**) to calculate the shortest path from the router to all destinations in the network.
- This results in a **routing table** that specifies the best path to each destination.

5. **Routing Table Update**:

- The router uses the SPF calculation to create and update its **Routing Table**, ensuring the best paths are used for packet forwarding.

6. **Periodic Updates and LSAs**:

- OSPF routers periodically exchange LSAs and update their routing tables. If a link goes down or if there's a topology change, the router recalculates the SPF and updates the routing table accordingly.

OSPF Area Structure

- **Areas**: OSPF uses a hierarchical structure where the network is divided into **areas**. This improves scalability by limiting the size of the LSDB in each area. The backbone area (Area 0) is the central area that connects all other areas.

OSPF Key Features:

- **Classless Routing**: OSPF supports **CIDR (Classless Inter-Domain Routing)** and provides routing of variable-length subnet masks.
- **Faster Convergence**: OSPF quickly adapts to network changes and converges faster than RIP (Routing Information Protocol).
- **LSA Types**: Different LSA types are used for different network topologies, such as Router LSAs, Network LSAs, Summary LSAs, and AS External LSAs.

OSPF Example:

- **Scenario**: Consider three routers (R1, R2, R3) in an OSPF network, each connected by a network link.
  - **R1 sends Hello packets** to R2 and R3, establishing neighbors relationships.
  - **R2 and R3 send back Hello packets** to R1, forming a two-way communication.
  - **LSAs are exchanged** between the routers, and each router builds its LSDB.
  - **R1 performs SPF calculations** to determine the best path to reach all destinations and updates its routing table.

Working of BGP (Border Gateway Protocol)

BGP (Border Gateway Protocol) is an inter-domain (inter-AS) routing protocol used to exchange routing information between different autonomous systems (ASes) on the internet. It is a path-vector protocol and is crucial for routing between different organizations or networks.

BGP Working Process

1. **Establishing a BGP Session**:

- BGP routers, known as **BGP peers**, establish a **TCP connection** (port 179) between them to exchange routing information.
- **Open messages** are exchanged to establish the BGP session, including information about the AS number, BGP version, and other parameters.

2. **Exchange of Routing Information**:

- Once the BGP session is established, routers begin exchanging **BGP Update messages**, which contain information about reachable IP prefixes and the **AS Path**.
- The AS Path is a list of ASes through which the route has passed, and it's used to prevent routing loops.

3. **BGP Routing Table**:

- Each BGP router maintains a **Routing Information Base (RIB)**, which contains all the routes it has learned from other BGP routers.
- These routes are stored along with the associated AS path, next-hop information, and other attributes (e.g., Local Preference, MED, and AS Path).

4. **Route Selection**:

- BGP routers evaluate the routes based on various attributes to determine the best path for each destination. The decision process follows several rules:
  - Prefer the route with the **longest prefix match**.
  - Prefer the route with the **lowest AS path length**.
  - Prefer routes with **lower MED** (Multi-Exit Discriminator) values.
  - Prefer routes with **higher Local Preference** values (used within an AS).

5. **Propagation of Updates**:

- After determining the best route, BGP routers propagate updates to their neighbors. If a router learns of a new route, it sends out an **Update message** containing the new information.

6. **Route Withdrawal**:

- When a route becomes unavailable or a better route is found, BGP sends a **Withdraw message** to inform other routers about the route removal.

7. **Maintaining the Session**:

- BGP routers maintain their session and periodically send **Keepalive messages** to ensure the connection remains active.

BGP Example:

- **Scenario**: Suppose AS 100 and AS 200 are connected via BGP, with BGP routers in each AS.
- **Router R1** in AS 100 learns that a prefix **10.0.0.0/24** is reachable via **Router R2** in AS 200.
- **Router R1** adds this information to its RIB, noting that the route has passed through AS 200 (AS Path: 200).
- **Router R1** uses BGP's path selection criteria to determine whether this route is the best one, considering factors like Local Preference or AS Path length.

- **Router R1** then advertises this route to its neighbors, allowing them to learn about the prefix.

BGP Key Features:

- **Path Vector Protocol**: BGP uses a path-vector approach where the complete AS path is recorded for each route.
- **Policy-Based Routing**: BGP allows network administrators to configure routing policies based on attributes like AS Path, Local Preference, or MED.
- **Scalable**: BGP is highly scalable and is used to manage routing across the global internet.

6a) Explain IPV6 datagram format. (10 Marks)

Defination-1 Marks

Explanation-6 Marks

Diagram-3 Marks

6a) An IPv6 datagram consists of two main parts:

1. **IPv6 Header**: Contains essential control information for packet routing.
2. **Payload**: The actual data being transmitted (e.g., application data, upper-layer protocol data like TCP or UDP).

IPv6 Datagram Header Format

The IPv6 header is **fixed-length** (40 bytes) and contains the following fields:

Field	bits)	Description
Version	4 bits	Indicates the IP version; for IPv6, it is set to 6.
Traffic Class	8 bits	Used for quality of service (QoS) management; differentiates the data packets based on priority.
Flow Label	20 bits	Used to identify packets that belong to the same flow (stream of packets with specific QoS requirements).
Payload Length	16 bits	The length of the payload (data) section, i.e., the data after the header (not including the header).
Next Header	8 bits	Identifies the type of the next layer protocol (e.g., TCP, UDP, ICMPv6).
Hop Limit	8 bits	Similar to TTL (Time to Live) in IPv4; it limits the number of hops (routers) a packet can travel before being discarded.
Source Address	128 bits	The 128-bit IPv6 address of the sender (source) of the packet.
Destination Address	128 bits	The 128-bit IPv6 address of the receiver (destination) of the packet.

IPv6 Datagram Header Detailed Breakdown

1. **Version (4 bits)**:
    - This field specifies the version of the Internet Protocol. For IPv6, it is always **6**.
2. **Traffic Class (8 bits)**:
    - Similar to the **Type of Service** field in IPv4, this field is used to specify the priority or class of service (CoS). It helps in defining packet precedence and service level for data transfer (e.g., for QoS policies).
3. **Flow Label (20 bits)**:
    - This is a new field introduced in IPv6 that allows for identifying packets belonging to the same flow. A flow is a set of packets that have the same source, destination, and traffic characteristics.
    - It can be used by routers to give preferential treatment to certain types of traffic (e.g., real-time data like voice or video).
4. **Payload Length (16 bits)**:
    - This field specifies the length of the payload, i.e., the data portion of the packet excluding the header. The maximum payload length can be **65,535 bytes**. If the payload exceeds this size, the **Jumbo Payload Option** is used, which allows larger payloads.
5. **Next Header (8 bits)**:
    - This field indicates the protocol used in the next layer (above the IP layer). It specifies whether the next layer is TCP, UDP, ICMPv6, or any other protocol (such as **IPv6 Extension Headers**).
        - Common values:
            - **6** = TCP
            - **17** = UDP
            - **58** = ICMPv6
            - **41** = IPv6 encapsulated in IPv4 (for tunnelling)
6. **Hop Limit (8 bits)**:
    - Similar to the **TTL (Time to Live)** field in IPv4, this field limits the number of hops (routers) a packet can make. Each router that forwards the packet decrements the hop limit. If the hop limit reaches **0**, the packet is discarded. This helps prevent packets from circulating endlessly due to routing loops.
7. **Source Address (128 bits)**:
    - This is the **IPv6 address** of the sender. It is 128 bits long and is written in hexadecimal notation (e.g., **2001:0db8:85a3:0000:0000:8a2e:0370:7334**).
8. **Destination Address (128 bits)**:

- This is the **IPv6 address** of the receiver. It is also 128 bits long and is similarly written in hexadecimal notation.

6b) Write a Dijkstra's algorithm to compute the shortest path through graph. (6 Marks)

Defination-1 Marks

Explanation-3 Marks

Diagram-1 Marks

Example-1 Marks

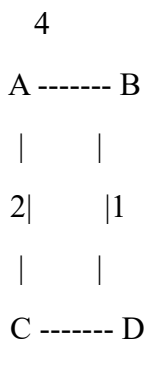
6b) Dijkstra's algorithm is a **greedy algorithm** that solves the **single-source shortest path problem** for a graph with non-negative edge weights. It finds the shortest path from a source vertex to all other vertices in the graph.

Steps of Dijkstra's Algorithm

1. **Initialization**:
   - Mark the distance to the source node as **0**.
   - Mark the distance to all other nodes as **infinity**.
   - Set the source node as the **current node**.
   - Set all nodes as unvisited.
2. **Visit the Current Node**:
   - For the current node, consider all of its unvisited neighbors.
   - Calculate their tentative distances from the source. The tentative distance is the sum of the distance from the source to the current node and the edge weight from the current node to the neighbor.
   - If the calculated tentative distance is smaller than the current recorded distance, update the shortest distance for the neighbor.
3. **Mark the Current Node as Visited**:
   - After considering all of the unvisited neighbors, mark the current node as visited. A visited node will not be checked again.
4. **Select the Next Node**:
   - Choose the unvisited node with the smallest tentative distance and make it the new **current node**.
   - Repeat steps 2-4 until all nodes are visited or the smallest tentative distance among the unvisited nodes is infinity (which means the remaining unvisited nodes are unreachable from the source).
5. **Result**:

- Once all nodes have been visited, the algorithm will have calculated the shortest path from the source to each node.

Consider the following graph with nodes labeled **A, B, C, D, E**:



- **Graph Representation** (edge weights between nodes)

```
graph = {
  'A': {'B': 4, 'C': 2},
  'B': {'A': 4, 'D': 1},
  'C': {'A': 2, 'D': 3},
  'D': {'B': 1, 'C': 3}
}
```

- **Start**: A

Step-by-Step Execution:

1. **Initialization**:
   - dist = {'A': 0, 'B': inf, 'C': inf, 'D': inf}
   - prev = {'A': None, 'B': None, 'C': None, 'D': None}
   - unvisited = ['A', 'B', 'C', 'D']
2. **Visit Node A**:
   - Tentative distances for neighbors of A:
     - B: 4 (A → B)
     - C: 2 (A → C)
   - Update dist and prev:
     - dist = {'A': 0, 'B': 4, 'C': 2, 'D': inf}
     - prev = {'A': None, 'B': 'A', 'C': 'A', 'D': None}

3. **Visit Node C (next minimum)**:
    - Tentative distances for neighbors of C:
        - A: 4 (C → A) (no update needed)
        - D: 5 (C → D)
    - Update dist and prev:
        - dist = {'A': 0, 'B': 4, 'C': 2, 'D': 5}
        - prev = {'A': None, 'B': 'A', 'C': 'A', 'D': 'C'}
4. **Visit Node B (next minimum)**:
    - Tentative distances for neighbors of B:
        - A: 8 (B → A) (no update needed)
        - D: 5 (B → D) (no update needed)
    - No updates.
5. **Visit Node D (last node)**:
    - No more updates needed as all nodes are visited.
6. **Result**:
    - dist = {'A': 0, 'B': 4, 'C': 2, 'D': 5}
    - prev = {'A': None, 'B': 'A', 'C': 'A', 'D': 'C'}

Reconstructing the Shortest Path

To find the shortest path from the source (A) to any node, trace the prep dictionary:

- **Path to B**: A → B
- **Path to C**: A → C
- **Path to D**: A → C → D

6c) Write a note on Routing information Protocol (RIP) algorithm. (4 Marks)

Defination-1 Marks

Explanation-3 Marks

6c) **Routing Information Protocol (RIP)** is one of the oldest distance-vector routing protocols used in **IP networks**. RIP helps routers determine the best route to reach a destination within a network. It is based on the **Bellman-Ford algorithm** and uses hop count as the metric to determine the shortest path to a destination.

RIP is typically used in **small to medium-sized networks** due to its simplicity, but it has limitations in scalability and speed. There are two main versions of RIP:

1. **RIP version 1 (RIP v1)**
2. **RIP version 2 (RIP v2)**

Basic Working of RIP

RIP is a **distance-vector protocol**, meaning each router sends information about its known routes to its neighbors periodically. Each router maintains a routing table, which lists the routes to all known destinations along with the **hop count** to reach each destination.

Key Characteristics of RIP:

- **Hop Count Metric**: RIP uses hop count as the metric for determining the best path. A hop is defined as the passage of data through one router.
    - The maximum number of hops allowed by RIP is **15**. This means any destination that requires more than 15 hops is considered **unreachable**.
- **Periodic Updates**: RIP routers send updates about their routing tables to their neighbors every **30 seconds**. This helps routers to maintain an up-to-date view of the network.
- **Routing Tables**: Each router in a RIP-based network maintains a routing table with the destination network and the corresponding **next-hop router** to reach that network, along with the number of hops.
- **Split Horizon and Poison Reverse**: These are techniques used to prevent **routing loops** and to improve the convergence time in RIP.
    - **Split Horizon**: A router will not advertise a route back to the router from which it learned the route.
    - **Poison Reverse**: A router advertises that a route has an infinite hop count (16 hops) if it is no longer valid, to avoid routing loops.

RIP Algorithm Process

1. **Initial State**:
    - Each router initializes its routing table with directly connected routes.
    - The hop count to a directly connected network is set to 1, and all other routes are set to infinity (16 hops).
2. **Periodic Updates**:
    - Every 30 seconds, routers exchange routing information with their neighbors. Each router sends its entire routing table to its neighbors.
    - The routers update their routing tables based on the received information. If a better route (with a lower hop count) is found, the routing table is updated accordingly.
3. **Convergence**:
    - RIP routers eventually reach a stable state where all routers in the network have the same routing information.

- The protocol tries to ensure that the network converges quickly by propagating changes in routing information.

4. **Route Failures**:
   - If a route becomes unreachable, the router will update its routing table by setting the hop count to 16 (infinity) for that route. This is the **count to infinity problem**, and techniques like **Split Horizon** and **Poison Reverse** are used to avoid loops.

RIP Versions

RIP v1 (Classful Routing)

- **Classful**: RIP v1 does not send subnet mask information in its routing updates, meaning it assumes all IP addresses belong to a classful network (A, B, or C).
- **Broadcast**: RIP v1 uses **broadcast** messages for updates, which can create unnecessary traffic in large networks.
- **No Authentication**: RIP v1 does not support authentication of routing updates, making it less secure.

RIP v2 (Classless Routing)

- **Classless**: RIP v2 supports **CIDR (Classless Inter-Domain Routing)** and sends the **subnet mask** along with the IP address, allowing more flexibility in network addressing.
- **Multicast**: RIP v2 uses **multicast** (address 224.0.0.9) for updates, which is more efficient than broadcasting.
- **Authentication**: RIP v2 supports **simple password authentication** to secure routing updates.

7a) Explain Go Back N Protocol working. (10 Marks)

Defination-1 Marks

Explanation-6 Marks

Diagram-2 Marks

Example-1 Marks

7a) The **Go-Back-N Protocol** is a sliding window protocol used in computer networks to ensure reliable data transmission over an unreliable network. It is commonly employed at the transport layer. This protocol manages **error detection**, **retransmission**, and **flow control**. Here's a detailed explanation of its working:

Key Components:

1. **Sender Window Size (N)**:
   The sender maintains a window of size **N** that determines how many frames can be sent before waiting for an acknowledgment (ACK).

2. **Receiver Window Size**:
   The receiver can only accept the next expected frame (size = 1). If a frame is out of order, it discards it and does not send an acknowledgment for it.
3. **Sequence Numbers**:
   Each frame is assigned a unique sequence number that wraps around after a certain range.

Working:

1. **Sending Data**:
   - The sender can transmit up to **N** frames without waiting for an acknowledgment.
   - Frames are sent continuously as long as the window is not full.
2. **Acknowledgments**:
   - The receiver sends a cumulative acknowledgment (ACK) for the last correctly received, in-order frame.
   - For example, if frames 0, 1, and 2 are received correctly, the receiver sends ACK for frame 2, meaning it has received all frames up to 2.
3. **Error Handling**:
   - If a frame is lost or an error occurs, the receiver discards all subsequent frames (out-of-order frames).
   - It sends no acknowledgment for the lost frame, which eventually causes the sender to detect a timeout.
4. **Retransmission**:
   - When the sender's timer for a specific frame expires (due to missing ACK), it retransmits **all frames starting from the unacknowledged frame**.
   - This is where "Go-Back-N" gets its name, as the sender goes back and retransmits N frames.

Example:

Parameters: Window size = 4

1. Sender sends frames 0, 1, 2, 3.
2. Receiver acknowledges frame 0 (ACK 0).
3. Frame 1 is lost during transmission.
4. Receiver discards frames 2 and 3 (out-of-order frames) and does not send ACK for them.
5. Sender times out waiting for ACK for frame 1.
6. Sender retransmits frames 1, 2, 3.

Pros:

- **Efficient use of bandwidth**: Allows sending multiple frames without waiting for individual ACKs.
- **Simpler receiver design**: Since the receiver only accepts in-order frames, its implementation is less complex.

Cons:

- **Wasted bandwidth on retransmission**: If a single frame is lost, all subsequent frames in the window are retransmitted, even if they were received correctly.
- **Not suitable for high-latency links**: Retransmitting many frames can lead to inefficiency.

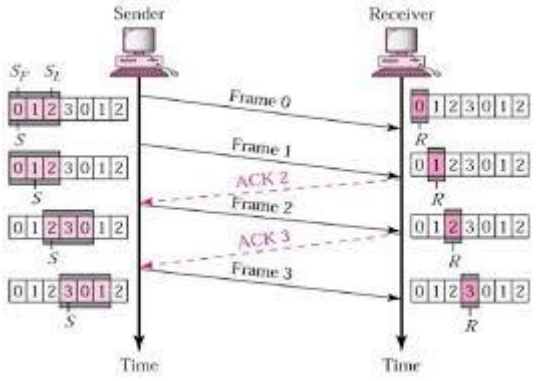


Figure: - Go-Back N Protocol

7b) With neat sketch, explain three-way handshaking of TCP connection establishment. (10 Marks)

Defination-1 Marks

Explanation-6 Marks

Diagram-2 Marks

Example-1 Marks

7b) The **three-way handshake** is the process used by the Transmission Control Protocol (TCP) to establish a reliable connection between a client and a server. It ensures that both parties are ready to communicate and that they agree on the sequence numbers for data transfer. Here's a detailed explanation:

Steps in the Three-Way Handshake:

1. **Step 1: SYN (Synchronization)**
    - The client initiates the connection by sending a **SYN (synchronize)** packet to the server.
    - This packet includes:
        - An **initial sequence number (ISN)** chosen by the client.

- A request to synchronize sequence numbers with the server.

Example:
Client → Server: SYN (ISN = X)

2. **Step 2: SYN-ACK (Synchronization + Acknowledgment)**
    - The server responds with a **SYN-ACK** packet.
    - This packet includes:
        - Its own **ISN (Y)** to synchronize sequence numbers with the client.
        - An **ACK** (acknowledgment) for the client's ISN, indicating it received the SYN packet.

Example:
Server → Client: SYN-ACK (ISN = Y, ACK = X + 1)

3. **Step 3: ACK (Acknowledgment)**
    - The client sends an **ACK** packet back to the server.
    - This packet acknowledges the server's ISN, confirming the connection is established.

Example:
Client → Server: ACK (ACK = Y + 1)

Final State:

- After the handshake, both the client and server are in a connected state, ready to exchange data.

Example Sequence:

- Client: SYN (ISN = 100)
- Server: SYN-ACK (ISN = 200, ACK = 101)
- Client: ACK (ACK = 201)

Key Features:

1. **Reliability**: Ensures both parties are ready for data transfer.
2. **Sequence Number Agreement**: Allows both sides to track the data flow and detect lost packets.
3. **Full-Duplex Communication Setup**: Establishes a bidirectional connection.

Importance:

- The three-way handshake is essential for establishing a reliable connection, avoiding issues like duplicate packets or connection mismatches.

8a) With an outline, explain selective repeat protocol.

8a) The **Selective Repeat (SR)** protocol is a sliding window protocol used for reliable data transfer. Unlike the Go-Back-N protocol, it retransmits only the frames that are lost or corrupted, not all subsequent frames. This makes it more efficient in handling errors, especially in networks with high latency or error rates.

Key Features of the Protocol:

1. **Sliding Window Mechanism**:
    - Both the sender and receiver maintain a window of size **N**.
    - The sender can transmit up to **N** frames without waiting for an acknowledgment.
    - The receiver can buffer out-of-order frames and acknowledge each frame individually.
2. **Acknowledgments (ACK)**:
    - The receiver sends individual acknowledgments for correctly received frames, even if they are out of order.
    - Cumulative acknowledgment is not used.
3. **Retransmissions**:
    - Only the frames that are reported as lost or corrupted are retransmitted.
    - This prevents unnecessary retransmission of correctly received frames.

Working of Selective Repeat Protocol:

1. **Sender Side**:
    - The sender transmits frames up to the window size.
    - It starts a timer for each frame sent.
    - If a frame's acknowledgment is not received before the timer expires, only that specific frame is retransmitted.
2. **Receiver Side**:
    - The receiver accepts frames within the receiver window, even if they are out of order.
    - It buffers out-of-order frames and delivers them to the application layer in the correct sequence once missing frames are received.
3. **Error Handling**:
    - If a frame is lost or corrupted, the receiver discards it and waits for the sender to retransmit it.
    - Acknowledgments are sent only for successfully received frames.

8b) List and explain various services provided by User Datagram Protocol (UDP). (10 Marks)

Defination-2 Marks

Services & explanation-8 Marks

8b) User Datagram Protocol (UDP) provides several services to support the transmission of data between applications. Here are the key services provided by UDP:

1. **Connectionless Communication**:
    - UDP is a connectionless protocol, meaning there is no need to establish a connection before sending data. Each packet (datagram) is sent independently, and there is no session or handshake process involved.
    - This makes UDP faster, but it comes at the cost of reliability and ordering.
2. **Unreliable Data Delivery**:
    - UDP does not guarantee delivery of packets. If a packet is lost during transmission, it is not retransmitted. The sender is not notified about lost packets, making UDP an unreliable protocol.
    - This is suitable for applications where speed is more critical than reliability, such as streaming or real-time applications.
3. **No Acknowledgments**:
    - Unlike Transmission Control Protocol (TCP), UDP does not require any acknowledgment from the receiver for the data sent. This reduces the overhead and increases transmission speed.
4. **No Flow Control**:
    - UDP does not implement flow control mechanisms to regulate the rate of data transmission between sender and receiver. It relies on the application to handle congestion or buffer overflow situations.
5. **No Error Recovery**:
    - UDP includes a checksum for error detection in the header of each datagram to check for integrity, but it does not offer any error recovery mechanisms. If a datagram is corrupted, it is discarded without any retransmission.
6. **Data Integrity (Checksum)**:
    - UDP uses a checksum to verify the integrity of the data in each datagram. This ensures that the data received by the destination is not corrupted during transmission, but there is no mechanism to fix it (other than dropping the datagram).
7. **Multiplexing**:
    - UDP allows multiple applications to use the same network interface simultaneously. It uses port numbers in its header to identify the sending and receiving applications, allowing multiplexing of communication on the same machine.

8. **Low Overhead**:
    - The UDP header is simple and small, containing only the source port, destination port, length, and checksum. This makes it lightweight compared to TCP, which has more fields and requires more processing.
9. **Broadcast and Multicast**:
    - UDP supports broadcasting and multicasting, which allows data to be sent to multiple receivers in a single transmission. This is useful in applications like video conferencing, live streaming, and DNS queries.
10. **Faster Data Transmission**:

- Due to its minimalistic design, UDP has lower latency compared to TCP, which is suitable for time-sensitive applications like VoIP, online gaming, and video streaming, where a slight delay in transmission could degrade the quality of service.

9a) Briefly explain Secure Shell (SSH). (10 Marks)

Defination-2 Marks

Explanation-8 Marks

9a) Secure Shell (SSH) is a cryptographic network protocol used to securely access and manage devices over an unsecured network, such as the internet. It provides a secure channel for communication between a client and a server, ensuring confidentiality, integrity, and authenticity of the transmitted data. Here's a brief overview of its key features:

1. **Secure Remote Login**:
    - SSH allows users to log into remote systems securely, replacing older, less secure protocols like Telnet or rlogin. It encrypts the entire session, preventing unauthorized interception of data.
2. **Encryption**:
    - SSH uses strong encryption techniques (such as AES or RSA) to ensure that the data exchanged between the client and server remains confidential and is not readable by attackers.
3. **Authentication**:
    - SSH supports multiple forms of authentication, including password-based and public-key authentication. Public-key authentication provides a more secure way of authenticating, as it uses a pair of cryptographic keys (public and private) rather than relying on passwords.
4. **Data Integrity**:
    - SSH ensures data integrity by using cryptographic hash functions. This prevents data from being altered during transmission.
5. **Secure File Transfer**:

- SSH also supports secure file transfer protocols like SFTP (Secure File Transfer Protocol) and SCP (Secure Copy Protocol), enabling secure transfer of files between systems.

6. **Port Forwarding**:
    - SSH can tunnel other protocols through its secure connection via port forwarding, allowing encrypted communication for otherwise insecure protocols, such as HTTP or VNC.

7. **Versatility**:
    - SSH is commonly used for managing remote servers, transferring files, and securing network services, making it a versatile tool for system administrators and developers.

Due to these features, SSH is widely used for remote administration, secure file transfers, and protecting sensitive data communications over untrusted networks.

9b) Write a note on Request message & response message formats of HTTPs. (10 Marks)

Request Message- 5 Marks

Response Message- 5 Marks

9b) In HTTPS (Hypertext Transfer Protocol Secure), data is exchanged securely between a client (usually a web browser) and a server. HTTPS uses encryption, typically with SSL/TLS, to ensure the confidentiality and integrity of the messages being exchanged. Here's a breakdown of the request and response message formats in HTTPS:

1. HTTPS Request Message Format

An HTTPS request message is sent from the client to the server to request resources. The request message consists of several components:

- **Request Line**:
    - **Method**: Specifies the type of action (GET, POST, PUT, DELETE, etc.).
    - **URI (Uniform Resource Identifier)**: The path to the resource being requested (e.g., /index.html).
    - **HTTP Version**: The version of HTTP being used

Headers: Headers provide additional information about the request. Some common headers include:

- **Host**: Specifies the domain name of the server (e.g., Host: www.example.com).
- **User-Agent**: Identifies the client (browser or application) making the request (e.g., User-Agent: Mozilla/5.0).

- **Accept**: Specifies the types of data the client can process (e.g., Accept: text/html).
- **Connection**: Defines whether the connection should be kept open or closed after the request

Body (Optional): In methods like POST or PUT, the body contains data sent to the server (e.g., form data, JSON payload). For GET requests, the body is typically empty.

HTTPS Response Message Format

An HTTPS response message is sent from the server to the client to provide the requested resource or a status update. The response message consists of the following components:

- **Status Line**:
    - **HTTP Version**: The version of HTTP being used (e.g., HTTP/1.1).
    - **Status Code**: A numeric code indicating the result of the request (e.g., 200 for success, 404 for "Not Found").
    - **Reason Phrase**: A brief textual description corresponding to the status code

Headers: Response headers provide metadata about the response. Some common response headers include:

- **Content-Type**: Indicates the media type of the response body (e.g., Content-Type: text/html).
- **Content-Length**: Specifies the length of the response body in bytes (e.g., Content-Length: 348).
- **Server**: Identifies the server software (e.g., Server: Apache/2.4).
- **Date**: The date and time the response was sent (e.g., Date: Wed, 24 Jan 2025 12:00:00 GMT).

Body: The body of the response contains the actual content requested by the client. For example, it might contain HTML, JSON, or images, depending on the type of request and the server's response.

10a) With neat diagram, explain the basic model of FTP. (4 Marks)

Explanation-3 Marks

Diagram-1 Marks

10a) FTP (File Transfer Protocol) is used for transferring files between a client and a server over a TCP/IP network. It follows a client-server model, where the client requests files or services, and the server provides them. The FTP model typically involves two main connections between the client and the server:

1. **Control Connection (Command Channel)**: Used for sending commands and receiving responses.
2. **Data Connection**: Used for transferring the actual file data.

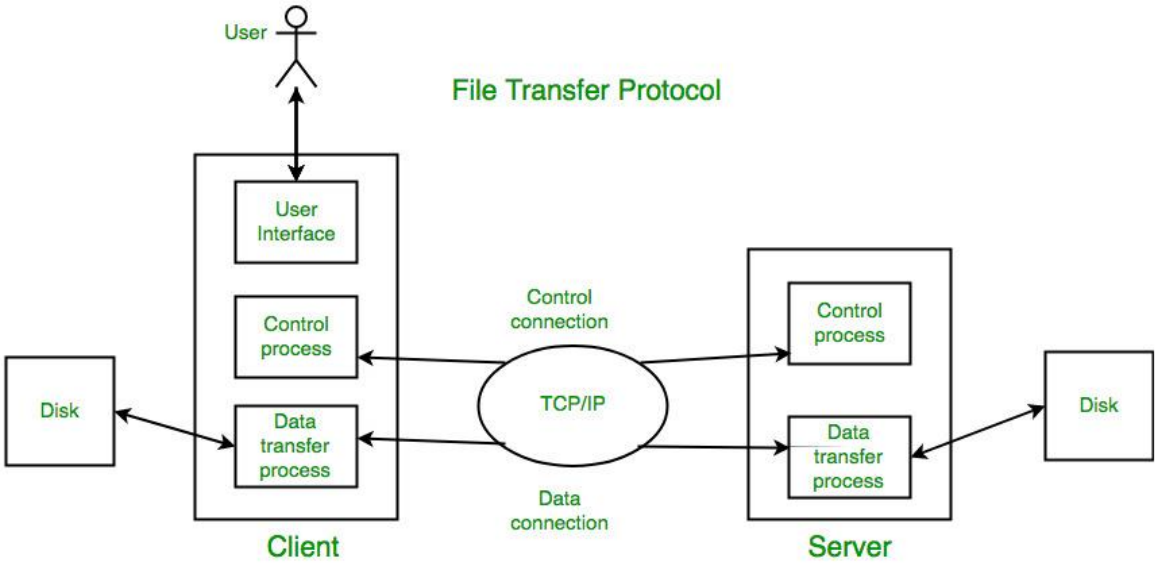


Figure: - File Transfer Protocol

Components of the FTP Model:

1. **Control Connection (Port 21)**:
    - **Purpose**: This is the channel through which the client and server communicate for control purposes. It is used to send FTP commands (like login, file commands) and receive responses.
    - **Communication**: The client opens the connection to port 21 of the server, where it sends commands such as USER, PASS, LIST, RETR, and so on. The server responds with status codes and messages.
2. **Data Connection (Port 20 or Passive Mode)**:
    - **Purpose**: This channel is used for transferring actual file data between the client and server.
    - **Communication**:
        - **Active Mode**: The client establishes the control connection on port 21, and the server opens a data connection from port 20 to the client. This method is less common due to firewall restrictions.
        - **Passive Mode**: The server opens a random port and tells the client to connect to it for data transfer. This mode is more common, especially when the client is behind a firewall or NAT (Network Address Translation).

Flow of FTP Communication:

1. **Connection Establishment**:

- The client opens a control connection to the server on port 21 and sends the login credentials (username and password) for authentication.

2. **Command and Response**:
   - The client sends FTP commands like LIST, RETR (retrieve file), STOR (store file), etc., through the control connection.
   - The server responds with status codes (e.g., 200 OK, 550 File Not Found).
3. **Data Transfer**:
   - Once the command is processed (e.g., RETR for downloading a file), a data connection is established either in active or passive mode.
   - The file is transferred over this data connection.
4. **Termination**:
   - Once the file transfer is complete, the data connection is closed, and the control connection can also be terminated when the session ends.

10b) Describe the architecture of electronic mail(email). (6 Marks)

Defination-1 Marks

Explanation-4 Marks

Diagram-1 Marks

10b) The architecture of email involves several components that work together to ensure the proper delivery, receipt, and management of messages. Email follows a **client-server model**, and the entire process can be broken down into key components and protocols used for sending, receiving, and storing email.

1. Email Clients (User Interface)

- **Purpose**: Email clients are the software applications or programs used by end users to send, receive, and manage email messages. They can be either **desktop-based** (like Microsoft Outlook, Thunderbird) or **web-based** (like Gmail, Yahoo Mail).
- **Responsibilities**:
  - Composing and reading emails
  - Managing inbox and folders
  - Sending attachments
  - Syncing email data with the server

2. Mail Servers (Email Servers)

- **Purpose**: Mail servers are responsible for handling the email messages sent and received by users. There are typically two types of mail servers involved:
  - **Incoming Mail Servers** (for receiving messages)

- **Outgoing Mail Servers** (for sending messages)

Incoming Mail Servers:

These servers are responsible for receiving and storing emails for users. They follow protocols such as:

- **POP3 (Post Office Protocol v3)**:
  - Used for retrieving emails from the server. POP3 downloads emails from the server and stores them on the client's device. It typically deletes the email from the server after download.
- **IMAP (Internet Message Access Protocol)**:
  - Used for retrieving emails while keeping them stored on the server. IMAP allows users to access their emails from multiple devices without deleting the messages from the server, providing a more flexible way of managing emails.

Outgoing Mail Servers:

These servers are responsible for sending email messages from clients to recipients. They use protocols like:

- **SMTP (Simple Mail Transfer Protocol)**:
  - The protocol used to send outgoing emails. SMTP is responsible for the delivery of email messages between servers. It works by relaying email to the destination server using the recipient's domain.

3. Mail Transfer Agents (MTAs)

- **Purpose**: MTAs are responsible for transferring email messages between mail servers. When you send an email, the email client contacts the **SMTP server** (outgoing mail server). The SMTP server then forwards the message to the destination server (recipient's mail server) using a series of MTAs.
- **Responsibilities**:
  - Relaying email from the sender's email server to the recipient's email server.
  - Ensuring the correct routing of the email message to the recipient's server.

4. Mail Delivery Agents (MDAs)

- **Purpose**: MDAs are responsible for the final delivery of the email to the recipient's mailbox on the incoming mail server.
- **Responsibilities**:
  - After the MTA forwards the email to the recipient's server, the MDA delivers the email to the user's mailbox.
  - Examples include **Dovecot** and **Procmail**.

5. Email Protocols

The following protocols are essential for the functioning of email:

- **SMTP (Simple Mail Transfer Protocol)**:
    - Used for sending emails from the sender's email client to the email server and between email servers (to deliver the message to the recipient's server).
- **POP3 (Post Office Protocol v3)**:
    - Used for retrieving emails from the server. POP3 downloads emails to the client and removes them from the server, so they cannot be accessed from another device.
- **IMAP (Internet Message Access Protocol)**:
    - Used for retrieving emails from the server while keeping them on the server. IMAP allows users to organize and manage their emails from different devices and retains the emails on the server for synchronization.

6. Email Addressing

- **Email Address**: A unique identifier for an email recipient, typically in the form of username@domain.com. The username represents the individual recipient, while the domain identifies the mail server.
- **Example**: john.doe@example.com

7. Email Flow (Sending and Receiving Process)

The process of sending and receiving email can be broken down into the following steps:

Sending an Email:

1. The user creates and sends an email via an email client (e.g., Outlook or Gmail).
2. The email client contacts the **SMTP server** (outgoing mail server) using SMTP.
3. The SMTP server checks the recipient's domain and finds the appropriate **MTA**.
4. The email is passed through the MTA to the recipient's **incoming mail server**.
5. The email is then handed over to the **MDA** (Mail Delivery Agent), which places it in the recipient's mailbox.

Receiving an Email:

1. The recipient's email client contacts the **incoming mail server** using either **POP3** or **IMAP** to check for new messages.
2. The mail server retrieves the email from the user's mailbox and forwards it to the client.
3. The email client displays the message to the recipient.

8. Security Measures in Email

Email systems often implement security measures to protect the integrity and confidentiality of email communication:

- **TLS (Transport Layer Security)**: Ensures encrypted communication between mail clients and servers.
- **SPF (Sender Policy Framework)**: Helps verify the sender's domain to prevent email spoofing.
- **DKIM (DomainKeys Identified Mail)**: Uses cryptographic signatures to verify the authenticity of the sender's domain.
- **DMARC (Domain-based Message Authentication, Reporting & Conformance)**: A policy framework that works with SPF and DKIM to prevent email abuse and phishing.

10c) Briefly explain Recursive Resolution & Iterative Resolution in DNS. (10 Marks)

Explanation-5+5 Marks

10c) DNS (Domain Name System) is used to translate human-readable domain names (like www.example.com) into IP addresses that computers can use to communicate with each other. When a client (e.g., a web browser) needs to resolve a domain name, the DNS query can be processed in one of two ways: **recursive resolution** or **iterative resolution**.

1. Recursive Resolution:

- **Definition**: In recursive resolution, the DNS client (usually a resolver) sends a query to a DNS server, and the server is responsible for fully resolving the query by either returning the final answer (IP address) or querying other DNS servers on behalf of the client until the answer is found.
- **Process**:
    - The client sends a query to a recursive DNS resolver.
    - If the resolver does not have the answer in its cache, it queries other DNS servers, starting from the **root server** to the **TLD (Top-Level Domain) server**, and then to the **authoritative name server** for the domain.
    - The recursive DNS server continues to query servers until it gets the final IP address and returns it to the client.
- **Key Points**:
    - The client sends only one request, and the server handles the entire process.
    - The server must perform all the steps to resolve the query.
    - It's slower than iterative resolution but provides a complete answer to the client.

2. Iterative Resolution:

- **Definition**: In iterative resolution, the DNS client queries a DNS server, and the server provides the best answer it has (which may not be the final answer). If the server doesn't know the answer, it provides a referral to another server that may know more.

- **Process**:
  - The client sends a query to a DNS resolver.
  - If the resolver does not know the answer, it sends a referral to a server that is closer to the final answer (e.g., a root server or a TLD server).
  - The client then sends the query to the next server (according to the referral) and repeats the process until it receives the final IP address.
- **Key Points**:
  - The client must follow the referrals and may have to send multiple queries to different servers.
  - The server only provides partial answers or referrals to other servers.
  - It’s faster than recursive resolution, but the client is more involved in the resolution process.