
Sub:	**Computer Networks**					Sub Code:	**BCS502**	Branch:	AIML/CSE(AIML)
Date:	**17/12/24**	Duration:	90 min's	Max Marks:	50	Sem/Sec:	V A		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1 a)	Explain DNS Name Space, DNS in the internet and resolution.	10M	CO3	L2
2 a)	Explain UDP services along with neat diagram Pseudo header for checksum.	10M	CO2	L2
3 a)	Explain local versus remote logging in TELNET.	5M	CO3	L2
3 b)	With a neat diagram, explain how SMTP can be used for transmitting mails from sender to receiver.	5M	CO3	L2
4 a)	Explain following with diagram. 1. WWW 2. HTTP 3. FTP	10M	CO3	L2
5 a)	Draw TCP Segment Structure. Describe the various fields of TCP segment Structure	10M	CO2	L2

1 a) Explain DNS Name Space, DNS in the internet and resolution.

DNS Name Space

DNS (Domain Name System) provides a hierarchical structure for naming resources on the internet. The DNS namespace is structured as a tree, with the following characteristics:

1. **Root Node**:
   - The topmost level of the hierarchy is represented as a dot (.).
   - It serves as the entry point to the DNS namespace.
2. **Top-Level Domains (TLDs)**:
   - The first level below the root node.
   - Examples include .com, .org, .edu, .gov, .net, and country-specific TLDs like .in, .uk, etc.
3. **Second-Level Domains**:
   - Directly below TLDs, e.g., example.com or university.edu.
   - These domains are typically registered by organizations or individuals.
4. **Subdomains**:
   - Levels below second-level domains, e.g., mail.example.com or research.university.edu.

5. **Hostnames**:
    - Specific machines or resources in the hierarchy, e.g., server1.research.university.edu.

DNS in the Internet

The Domain Name System acts as the internet's directory. It translates human-readable domain names (like www.google.com) into machine-readable IP addresses (like 142.250.183.100).

- **Distributed Architecture**: DNS is distributed across a network of servers globally to ensure scalability and reliability.
- **Components**:
    1. **DNS Servers**:
        - **Root Servers**: Handle requests for TLDs.
        - **TLD Servers**: Direct requests to second-level domains.
        - **Authoritative Name Servers**: Provide answers for specific domains.
    2. **Resolvers**: Local DNS clients or ISPs that query the DNS servers on behalf of users.

DNS Resolution

DNS resolution is the process of translating a domain name into an IP address. It involves the following steps:

1. **Request Initiation**:
    - A user enters a domain name in a browser. The request is sent to a local resolver.
2. **Recursive Query**:
    - The resolver contacts a root DNS server to find the TLD server for the requested domain.
3. **Iterative Query**:
    - The root server provides the address of the appropriate TLD server.
    - The resolver then queries the TLD server for the domain's authoritative name server.
4. **Authoritative Response**:
    - The authoritative server responds with the IP address of the requested domain.
5. **Response to Client**:
    - The resolver sends the resolved IP address to the user's browser, which connects to the desired resource.

Example:

When resolving www.example.com:

1. The resolver queries the root server, which points to the .com TLD server.
2. The .com server points to the authoritative server for example.com.
3. The authoritative server provides the IP address of www.example.com.
4. The browser uses the IP address to establish a connection to the website.

This hierarchical and distributed design makes DNS efficient and scalable for the entire internet.

2 a) Explain UDP services along with neat diagram Pseudo header for checksum.

UDP Services

The **User Datagram Protocol (UDP)** is a connectionless and lightweight transport layer protocol used for sending and receiving data over the network. Unlike TCP, UDP provides minimal error recovery and does not ensure reliable delivery, making it suitable for applications that require speed over reliability.

Key UDP Services

1. **Connectionless Communication**:
    - No setup or teardown is needed for communication.
    - Data is sent directly from sender to receiver without establishing a connection.
2. **Message-Oriented Communication**:
    - Preserves data boundaries; messages are sent and received as independent units.
    - Each datagram is treated as a separate packet.
3. **Unreliable Delivery**:
    - No acknowledgment of data receipt.
    - Lost or out-of-order packets are not retransmitted or reordered.
4. **Fast Transmission**:
    - Minimal overhead compared to TCP.
    - Used in applications like video streaming, VoIP, and online gaming where speed is critical.
5. **Checksum for Error Detection**:
    - Includes an optional checksum to detect errors in transmitted data.
6. **Multiplexing and Demultiplexing**:
    - Uses port numbers to distinguish different applications on the same host.

UDP Packet Structure

A UDP packet consists of the following fields:

Field Name	Size (bits)	Description
Source Port	16	Identifies the sending application.
Destination Port	16	Identifies the receiving application.
Length	16	Total size of the UDP header and data in bytes.
Checksum	16	Used for error detection.
Data	Variable	Application-specific data.

Pseudo Header for UDP Checksum

The checksum is calculated using a **pseudo-header**, which includes additional information to ensure end-to-end data integrity. The pseudo-header is not transmitted but is used during checksum calculation.

Pseudo Header Structure

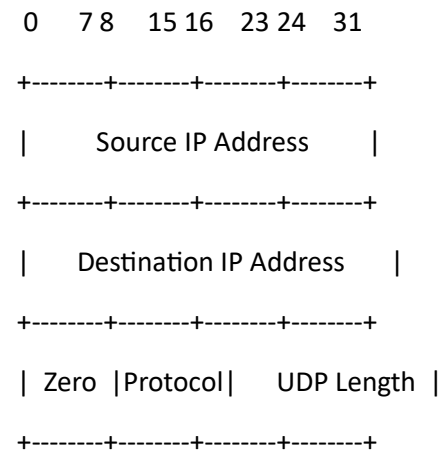
Field Name	Size (bits)	Description
Source IP Address	32	IP address of the sender.
Destination IP Address	32	IP address of the receiver.
Protocol	8	Indicates the protocol (17 for UDP).
UDP Length	16	Total length of the UDP packet (header + data).

Steps to Calculate Checksum

1. Construct the pseudo-header.
2. Append the UDP header and data to the pseudo-header.
3. Divide the entire header and data into 16-bit words.
4. Calculate the one's complement sum of these 16-bit words.
5. Perform the one's complement of the final sum to get the checksum.

Diagram: UDP Pseudo Header

Below is a visual representation of the UDP pseudo-header:



Applications of UDP

- **Streaming Media**: Video/audio streaming where delays matter more than lost data.
- **Online Gaming**: Fast-paced games prioritize speed over data reliability.
- **DNS Queries**: Quick and simple queries for domain name resolution.
- **IoT Devices**: Lightweight communication in resource-constrained systems.

UDP is ideal for real-time applications where speed and efficiency outweigh reliability.

2 a) Explain UDP services along with neat diagram Pseudo header for checksum. 10M CO2 L2

UDP Services

The **User Datagram Protocol (UDP)** is a connectionless and lightweight transport layer protocol used for fast and efficient data transfer. Unlike TCP, UDP does not guarantee reliability, order, or data integrity beyond error detection.

Features of UDP Services

1. **Connectionless Communication**:
    - No handshake is required to establish or terminate the connection.
    - Data is sent immediately, reducing latency.
2. **Message-Oriented Communication**:
    - Each UDP packet (datagram) is sent independently and retains its boundaries.
3. **Unreliable Data Transfer**:
    - No acknowledgment, retransmission, or flow control.
    - Packets may be lost, duplicated, or arrive out of order.
4. **Checksum for Error Detection**:
    - Provides basic error detection for data integrity using a checksum.

    - Errors in transmission can be detected but not corrected.
5. **Multiplexing and Demultiplexing**:
    - Uses port numbers to distinguish multiple applications running on a single host.
6. **Low Overhead**:
    - Faster than TCP due to reduced protocol overhead.

Applications of UDP

- **Real-Time Applications**: Video conferencing, VoIP, and live streaming.
- **Gaming**: Online multiplayer games prioritize speed over reliability.
- **DNS**: Domain Name System queries use UDP for quick lookups.
- **IoT Systems**: Suitable for lightweight communications in constrained environments.

UDP Packet Structure

Field Name	Size (bits)	Description
Source Port	16	Identifies the sending application.
Destination Port	16	Identifies the receiving application.
Length	16	Total size of UDP header and data in bytes.
Checksum	16	Provides error detection for the header and data.
Data (Payload)	Variable	Application-specific data.

Pseudo Header for Checksum

The checksum ensures the integrity of the data during transmission. The UDP checksum is calculated using a **pseudo-header**, which includes parts of the IP header along with the UDP header and data. The pseudo-header is not transmitted but is used in the checksum calculation.

Structure of Pseudo Header

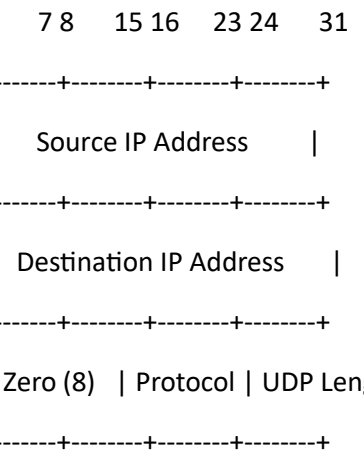
Field Name	Size (bits)	Description
Source IP Address	32	The sender's IP address.
Destination IP Address	32	The receiver's IP address.
Reserved (Zero)	8	Set to 0.
Protocol	8	Protocol number (17 for UDP).

Field Name	Size (bits)	Description
UDP Length	16	Length of the UDP packet (header + data).

Checksum Calculation Steps

1. Construct the pseudo-header.
2. Append the UDP header and payload to the pseudo-header.
3. Divide the entire structure into 16-bit words.
4. Calculate the one's complement sum of all 16-bit words.
5. Perform the one's complement of the sum to generate the checksum.

Diagram: Pseudo Header



Advantages of UDP

- Minimal latency due to low overhead.
- Ideal for real-time applications.
- Simplicity in implementation.

Limitations of UDP

- Lack of reliability and sequencing.
- No congestion control mechanisms.

UDP provides a fast and efficient transport mechanism where reliability is handled by the application layer, if necessary.

4o

You said:

3 b) With a neat diagram, explain how SMTP can be used for transmitting mails from sender to receiver. 5M CO3 L2

SMTP (Simple Mail Transfer Protocol)

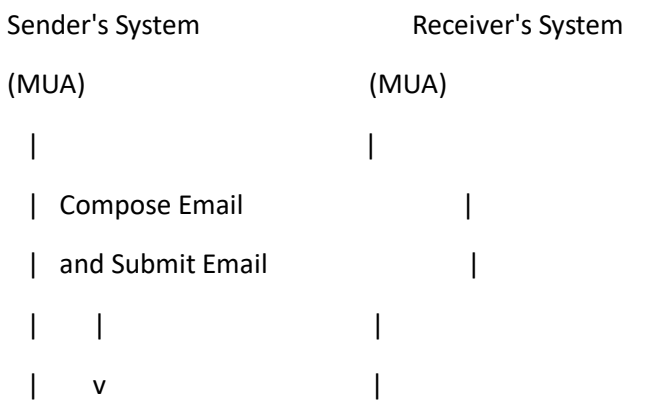
SMTP is a protocol used for sending emails across networks. It operates at the application layer and uses a client-server model to transfer messages between a sender's and a receiver's email servers. SMTP is connection-oriented and relies on TCP for reliable delivery.

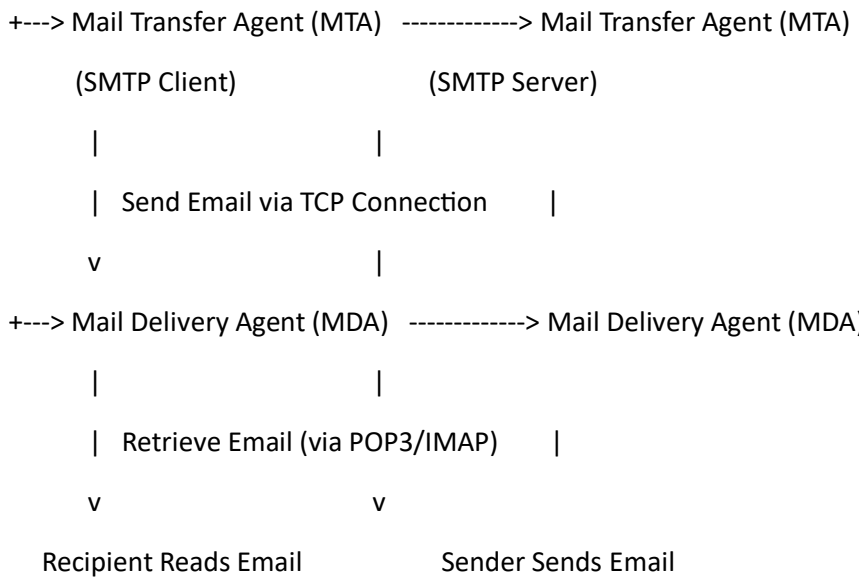
Steps in SMTP for Email Transmission

SMTP facilitates mail transmission through the following stages:

1. **Mail Submission**:
    - The sender composes the email using a Mail User Agent (MUA), such as Outlook or Gmail.
    - The email is handed over to the sender's Mail Transfer Agent (MTA) for delivery.
2. **Mail Routing**:
    - The sender's MTA establishes a TCP connection with the receiver's MTA (usually port 25 for SMTP).
    - The email is transmitted using the SMTP commands (e.g., HELO, MAIL FROM, RCPT TO, DATA, QUIT).
3. **Mail Storage on Receiver's Server**:
    - The receiver's MTA stores the email in the Mail Delivery Agent (MDA), like POP3 or IMAP servers.
    - The MDA manages the email for retrieval by the recipient.
4. **Mail Retrieval**:
    - The recipient uses their MUA to fetch the email from the MDA using POP3 or IMAP protocols.

Diagram: SMTP Email Transmission





SMTP Commands

1. **HELO/EHLO**: Introduces the client to the server.
2. **MAIL FROM**: Specifies the sender's email address.
3. **RCPT TO**: Specifies the recipient's email address.
4. **DATA**: Begins the transmission of the email content.
5. **QUIT**: Terminates the session.

Advantages of SMTP

- Reliable delivery of emails.
- Simple and widely used protocol.
- Supports multiple recipients in a single transaction.

Limitations of SMTP

- Does not support multimedia attachments directly (handled by MIME).
- Relies on plain-text communication, which can be insecure without encryption like STARTTLS or SSL/TLS.

SMTP remains the backbone of email communication, working seamlessly with additional protocols like POP3 and IMAP for end-to-end email delivery.

4 a) Explain following with diagram. 1. WWW 2. HTTP 3. FTP 10M CO3 L2

1. WWW (World Wide Web)

The **World Wide Web (WWW)** is a system of interlinked hypertext documents and multimedia resources accessed via the Internet. It uses web browsers to retrieve and display content, enabling users to navigate between pages using hyperlinks.

Key Features:

- **Hypertext**: Links text and multimedia for navigation.
- **Uniform Resource Locator (URL)**: Used to identify resources on the web.
- **Protocols**: Relies on HTTP/HTTPS for communication between clients (browsers) and servers.

Architecture of WWW:

rust

Copy code

User <--> Browser <--> Internet <--> Web Server <--> Resources (HTML, CSS, Images)

Steps:

1. User enters a URL in the browser.
2. The browser sends an HTTP request to the web server.
3. The server responds with the requested resource.
4. The browser renders the content for the user.

---

2. HTTP (Hypertext Transfer Protocol)

HTTP is the primary protocol for transferring hypertext resources on the WWW. It follows a client-server architecture where the client (browser) sends requests to the server, and the server responds with resources.

Features of HTTP:

1. **Stateless**: Each request is independent.
2. **Request-Response Model**:
    - **Request**: Sent by the client, including method (e.g., GET, POST), URL, and headers.
    - **Response**: Sent by the server, including status codes, headers, and the resource.
3. **Supports Multimedia**: Transfers HTML, images, audio, and video.

HTTP Workflow:

Client (Browser) --> HTTP Request --> Web Server

Web Server --> HTTP Response --> Client (Browser)

Diagram:

User --> Browser --> HTTP Request --> Server --> HTTP Response --> User

---

3. FTP (File Transfer Protocol)

FTP is a standard network protocol used to transfer files between a client and a server over a TCP-based network, such as the Internet.

Key Features:

1. **File Transfer**: Supports uploading and downloading files.
2. **Authentication**: Uses username and password for secure access.
3. **Active and Passive Modes**:
    - **Active**: The client opens a port for data transfer.
    - **Passive**: The server opens a port for data transfer.

FTP Workflow:

1. The client connects to the FTP server using a command-line interface or FTP client.
2. Authentication is performed using credentials.
3. Commands like GET (download) or PUT (upload) are executed to transfer files.

Diagram:

rust

Copy code

User <--> FTP Client <--> Internet <--> FTP Server <--> File System

---

Comparison and Applications

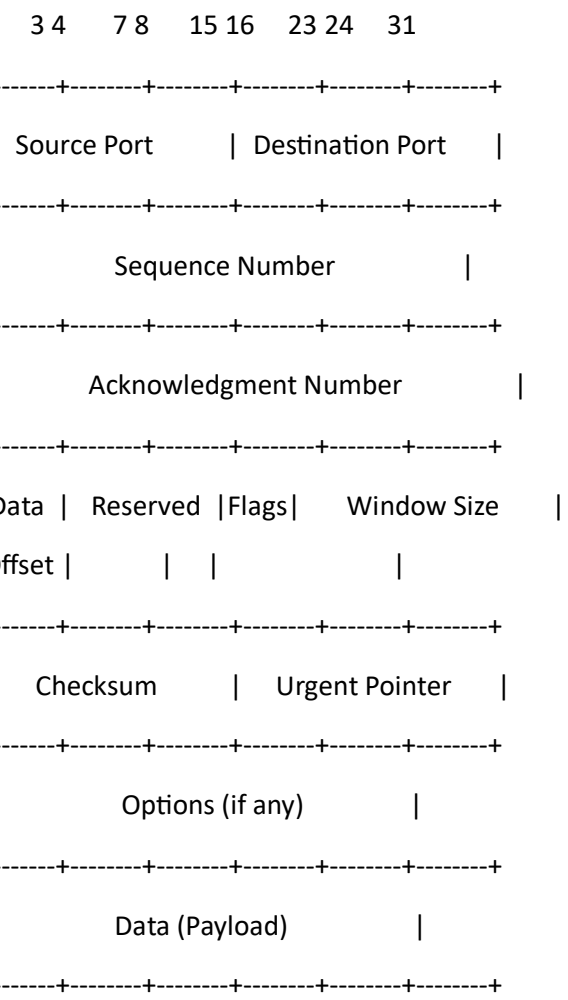
Protocol	Purpose	Features	Use Cases
WWW	Browsing web pages	Links hypertext with multimedia resources	Information access, browsing, online apps
HTTP	Transferring web resources	Request-response, stateless, supports caching	Websites, REST APIs
FTP	File transfer	Authentication, active/passive modes	File sharing, backups, hosting

5 a) Draw TCP Segment Structure. Describe the various fields of TCP segment Structure

TCP Segment Structure

The **TCP (Transmission Control Protocol)** segment structure is the basic unit of data transmission in the TCP protocol. It contains various fields that manage the delivery, sequencing, and reliability of data across a network. The TCP segment is encapsulated within an IP packet for transmission.

Diagram: TCP Segment Structure



Fields of the TCP Segment

1. Source Port (16 bits)

- Identifies the port number of the sender's application.
- Enables multiplexing and demultiplexing of applications.

2. Destination Port (16 bits)

- Identifies the port number of the receiving application.

3. Sequence Number (32 bits)

- Indicates the position of the first byte of the data in the segment within the overall data stream.

- Ensures proper ordering of data.

4. Acknowledgment Number (32 bits)

- Used by the receiver to inform the sender of the next expected byte in the sequence.
- Ensures reliable delivery by acknowledging received segments.

5. Data Offset (4 bits)

- Specifies the size of the TCP header in 32-bit words.
- Helps locate where the data begins.

6. Reserved (6 bits)

- Reserved for future use; set to zero.

7. Control Flags (6 bits):

Flags control the state and actions of the connection:

- **URG**: Urgent pointer field is valid.
- **ACK**: Acknowledgment number is valid.
- **PSH**: Push function; forces the receiver to deliver data immediately.
- **RST**: Resets the connection.
- **SYN**: Synchronizes sequence numbers during connection setup.
- **FIN**: Indicates no more data will be sent; used for connection termination.

8. Window Size (16 bits)

- Specifies the size of the sender's receive window (flow control mechanism).
- Indicates the number of bytes the sender is willing to accept.

9. Checksum (16 bits)

- Used for error detection on the entire TCP segment (header + data).
- Calculated using a pseudo-header that includes the IP addresses and protocol number.

10. Urgent Pointer (16 bits)

- Points to the end of the urgent data in the segment.
- Used when the URG flag is set.

11. Options (Variable length)

- Provides additional features like maximum segment size (MSS), window scaling, or timestamps.
- Length is variable and padded to ensure the header ends on a 32-bit boundary.

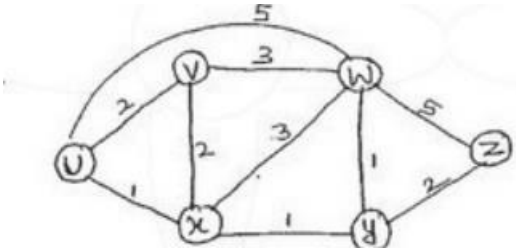
12. Data (Payload)

- The actual application data being transmitted.
- Length depends on the total segment size and the header length.

Key Features of the TCP Segment Structure

1. **Reliable Transmission**:
    - Sequence and acknowledgment numbers ensure data arrives in order and without loss.
2. **Error Detection**:
    - Checksum validates the integrity of data.
3. **Flow Control**:
    - Window size prevents overwhelming the receiver.
4. **Connection Control**:
    - Flags (SYN, FIN) manage connection setup and teardown.

Question 6.a Write Link State Routing Algorithm. Apply it to the following graph with source node as "U". Draw the least cost path tree and the forwarding table for node "U".



To solve this problem, I will provide a detailed explanation of **Link State Routing Algorithm (Dijkstra's Algorithm)**, apply it to the graph provided with source node U, draw the least cost path tree, and present the forwarding table for node U.

Link State Routing Algorithm (Dijkstra's Algorithm)

The steps are as follows:

1. **Initialization**:
    - Start from the source node (U).
    - Set the cost to the source node as 0 and to all other nodes as ∞ (infinity).
    - Mark all nodes as unvisited.

2. **Repeat**:
    - Pick the unvisited node with the smallest current cost.
    - Update the cost of its neighbors if a shorter path is found.
    - Mark the node as visited.
3. **Stop**:
    - Repeat until all nodes are visited.

Graph Analysis and Cost Table

The graph provided has:

- Nodes: U, V, W, X, Y, Z
- Edges with weights:
    - U ↔ V (2), U ↔ X (1)
    - V ↔ W (3), V ↔ X (2)
    - W ↔ Y (1), W ↔ Z (5)
    - X ↔ Y (1), X ↔ V (2)
    - Z ↔ Y (2)

The goal is to find the shortest path from the source node U to all other nodes.

Step-by-Step Application of Dijkstra's Algorithm

We proceed as follows:

Step	Current Node	Visited Nodes	Cost from U	Next Nodes to Check	Updated Costs
1	U	U	U: 0	V, X	V = 2, X = 1
2	X	U, X	X = 1	V, Y	V = 2 (no change), Y = 2
3	V	U, X, V	V = 2	W	W = 5
4	Y	U, X, V, Y	Y = 2	W, Z	W = 3, Z = 4
5	W	U, X, V, Y, W	W = 3	Z	Z = 4 (no change)
6	Z	All visited	Z = 4	None	Final Costs Calculated

Final Shortest Path Tree

- From the table, the least cost paths from node U are:

- **U → X**: Cost 1
- **U → V**: Cost 2
- **U → X → Y**: Cost 2
- **U → X → Y → W**: Cost 3
- **U → X → Y → Z**: Cost 4

The **tree structure** will look like this:

markdown

Copy code

```
    U
    |
    X
   / \
  Y   V
 / \
W   Z
```

Forwarding Table for Node U

Destination	Next Hop	Cost
V	V	2
W	X	3
X	X	1
Y	X	2
Z	X	4

Here:

- The **Next Hop** indicates the first node to which packets should be forwarded to reach the destination node.
- **Cost** is the shortest path cost from U to that destination.