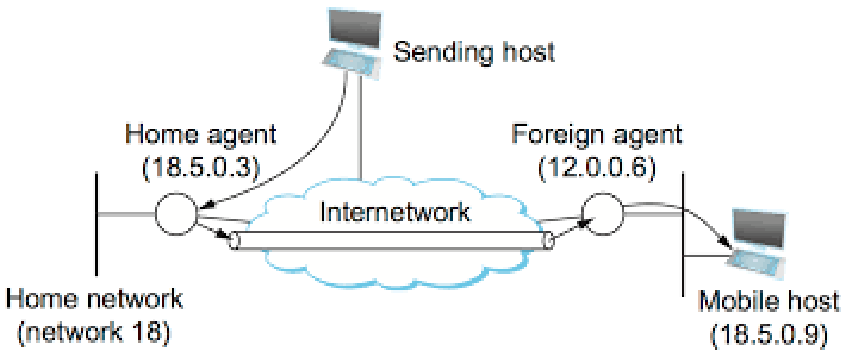
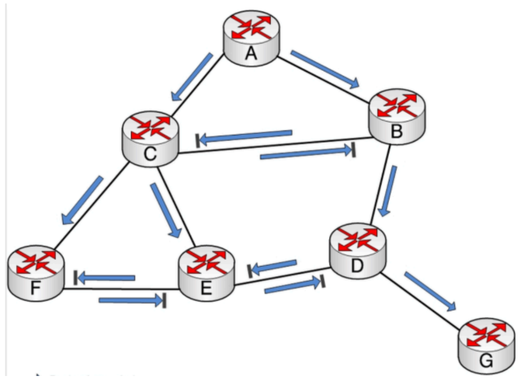
USN | | | | | | | | | | |

Internal Assessment Test III– March 2024

Sub:	Computer Networks				Sub Code:	21CS52	Branch:	CSE
Date:	14/03/2024	Duration:	90 mins	Max Marks: 50	Sem /Sec:	V / A, B & C Sec		

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1.	Consider the following network.  Sending host Home agent (18.5.0.3) Foreign agent (12.0.0.6) Internetwork Home network (network 18) Mobile host (18.5.0.9) The mobile host having IP 18.5.0.9 was initially connected to network 18 (as shown in the above given figure), connected to the Home Agent. Later it moves to a remote network and connected through foreign agent. Describe the steps, how a sending host can communicate with the mobile host?	10	CO2	L3
2.	What is broadcast routing. Explain Reverse Path Routing algorithm with reference to following Tree. 	10	CO2	L3
3.	What are Berkley Sockets? Explain Berkley Socket primitives used for TCP.	10	CO3	L2
4.	With the help of state sequence diagram, explain the 3 phases used in TCP Communication	10	CO3	L2
5.	Bob, a user, wants to access the homepage of a website called "www.example.com" using her web browser. Explain how the request message has been prepared by the browser and the response message by the server.	10	CO3	L3
6	Alice wants to access the website "www.example.com" from her web browser. Explain the various stages the client request needs to be processed w. r.t recursive and iterative approaches of DNS queries.	10	CO3	L3

CCI	CI	HOD

CO-PO and CO-PSO Mapping																			
Course Outcomes		**Bloms Level**	**Modules covered**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**	**PSO4**
CO1	Learn the basic needs of communication system.	**L1**	**1**	2	-	-	-	1	3	1	-	-	-	-	1	-	-	2	-
CO2	Interpret the communication challenges and its solution.	**L2,L3**	**1,2**	2	3	3	3	2	3	1	-	-	-	-	2	-	-	2	1
CO3	Identify and organize the communication system network components	**L3**	**1,2,3**	2	3	3	3	2	3	1	-	-	-	-	2	-	2	2	2
CO4	Design communication networks for user requirements.	**L3**	**1,2,3,4,5**	3	3	3	3	3	3	2	2	2	2	2	2	-	2	2	2

COGNITIVE LEVEL	REVISED BLOOMS TAXONOMY KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO)				CORRELATION LEVELS	
PO1	Engineering knowledge	PO7	Environment and sustainability	0	No Correlation
PO2	Problem analysis	PO8	Ethics	1	Slight/Low
PO3	Design/development of solutions	PO9	Individual and teamwork	2	Moderate/ Medium
PO4	Conduct investigations of complex problems	PO10	Communication	3	Substantial/ High
PO5	Modern tool usage	PO11	Project management and finance		
PO6	The Engineer and society	PO12	Life-long learning		
PSO1	Design and develop applications using different stacks of web and programming technologies.				
PSO2	Design and develop secure, parallel, distributed, networked, and digital systems.				
PSO3	Apply software engineering methods to design, develop, test and manage software systems.				
PSO4	Design and develop intelligent applications for business and industry.				

SOLUTION

1. To facilitate communication between the sending host and the mobile host after it has moved to a remote network and connected through a foreign agent, the following steps typically occur in Mobile IP (Internet Protocol) communication:

1. **Registration by the Mobile Host:** When the mobile host (18.5.0.9) moves to a remote network and connects through a foreign agent, it registers its new care-of address (COA) with its home agent. The COA is the temporary IP address assigned to the mobile host by the foreign agent.
2. **Binding Update to the Home Agent:** The mobile host sends a Binding Update message to its home agent, informing it of its new COA. This allows the home agent to update its binding cache, which keeps track of the mobile host's current location.
3. **Tunneling of Data Packets:** When the sending host wants to communicate with the mobile host, it sends packets destined for the mobile host's IP address (18.5.0.9). Since the home agent knows the COA of the mobile host, it encapsulates the packets destined for the mobile host within an IP-in-IP tunnel and forwards them to the COA.
4. **Delivery to the Foreign Agent:** The encapsulated packets are routed to the foreign agent serving the mobile host's current location based on the COA.
5. **Decapsulation and Delivery to the Mobile Host:** Upon receiving the encapsulated packets, the foreign agent decapsulates them, revealing the original IP packets destined for the mobile host. It then delivers these packets to the mobile host at its current location on the remote network.

By following these steps, the sending host can effectively communicate with the mobile host, even after it has moved to a remote network and connected through a foreign agent in a Mobile IP environment.

2. Broadcast Routing

Broadcast routing plays a role in computer networking and telecommunications. It involves transmitting data, messages, or signals from one source to destinations within a network. Unlike routing (one-to-one communication) or multicast routing (one-to-many communication) broadcast routing ensures that information reaches all devices or nodes within the network.Broadcast routing is a method used in computer networks to send a message from one node to all other nodes in the network. In broadcast routing, a source node sends a message to all other nodes, and each intermediate node forwards the message to all of its connected nodes until all nodes in the network receive the message.

3. Berkeley Sockets, often referred to simply as sockets, are a set of programming interfaces (APIs) and protocols used for establishing and managing network connections in a Unix-like operating system environment. They were initially developed at the University of California, Berkeley, hence the name "Berkeley Sockets."

These sockets provide a standardized way for applications to communicate over a network, whether it's a local area network (LAN), wide area network (WAN), or the internet. Sockets support various communication protocols, including Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Internet Protocol (IP).

For TCP communication using Berkeley Sockets, the following primitive functions are commonly used:

1. **socket():** This function creates a new socket that can be used for communication. It takes parameters specifying the communication domain (e.g., AF_INET for IPv4), socket type (e.g., SOCK_STREAM for TCP), and protocol (usually set to 0 for default protocol).
2. **bind():** The bind() function associates a socket with a specific network address, including the IP address and port number. This is necessary for servers to listen for incoming connections on a specific port.
3. **listen():** For server applications, the listen() function marks the socket as passive, allowing it to accept incoming connections from clients. It specifies the maximum number of pending connections that the socket's queue can hold.
4. **accept():** After a server socket is in the listening state, the accept() function is called to accept an incoming connection request from a client. It creates a new socket for communication with the client and returns the client's address information.
5. **connect():** In client applications, the connect() function initiates a connection to a remote server using the specified IP address and port number. Once the connection is established, the client can send and receive data with the server.

6. **send() and recv():** These functions are used for sending and receiving data over a connected TCP socket, respectively. The send() function sends data from the application to the remote peer, while recv() receives data from the remote peer into a buffer provided by the application.
7. **close():** The close() function closes a socket and releases any associated resources. It terminates the connection if it's a connected socket and prevents further communication.

These primitives provide a straightforward and powerful interface for building networked applications using TCP communication with Berkeley Sockets. They abstract away many of the complexities of network programming, allowing developers to focus on implementing the desired functionality.

4. TCP (Transmission Control Protocol) communication involves three main phases: connection establishment, data transfer, and connection termination. Let's illustrate each phase with a state sequence diagram:

Connection Establishment Phase:

```
Client                              Server
 |                                    |
 |        SYN (Sequence = x)           |
 |------------------------------------------>|
 |                                    |
 |     SYN-ACK (Sequence = y, ACK = x+1)  |
 |<------------------------------------------|
 |                                    |
 |             ACK (ACK = y+1)          |
 |------------------------------------------>|
 |                                    |
```

- The client initiates the connection by sending a SYN packet to the server, indicating its initial sequence number (x).
- The server responds with a SYN-ACK packet, acknowledging the client's sequence number (x) and sending its own sequence number (y).
- Finally, the client acknowledges the server's sequence number (y) with an ACK packet, and the connection is established.

Data Transfer Phase:

```
Client                              Server
 |                                    |
 |            Data Packet 1           |
 |------------------------------------------>|
 |                                    |
 |             ACK (ACK = x+n)          |
 |<-----------------------------------------|
 |                               |
 |          Data Packet 2              |
 |---------------------------------------->|
 |                               |
 |           ACK (ACK = x+n+m)           |
 |<-----------------------------------------|
 |                               |
```

- Once the connection is established, data packets can be exchanged between the client and server.
- The client sends data packets to the server, and the server acknowledges each packet by sending an ACK packet with the next expected sequence number.
- This process continues until all data is transmitted.

Connection Termination Phase:

```
  Client                          Server
   |                               |
   |          FIN (Sequence = x)          |
   |---------------------------------------->|
   |                               |
   |      ACK (ACK = x+1)                 |
   |<-----------------------------------------|
   |                               |
   |      FIN (Sequence = y)             |
   |<-----------------------------------------|
   |                               |
   |      ACK (ACK = y+1)                 |
   |---------------------------------------->|
   |                               |
```

- Either the client or server can initiate the connection termination by sending a FIN packet.
- The receiving end responds with an ACK to acknowledge the FIN packet.
- After both sides have sent and acknowledged the FIN packets, the connection is fully closed.

These state sequence diagrams illustrate the three phases of TCP communication: connection establishment, data transfer, and connection termination. Each phase involves specific packet exchanges between the client and server to establish, transfer, and terminate the connection.

5. When Bob, the user, wants to access the homepage of the website "www.example.com" using her web browser, the following process occurs:

Request Message Preparation by the Browser:

- Bob's web browser (e.g., Chrome, Firefox) prepares an HTTP (Hypertext Transfer Protocol) request message to fetch the homepage from "www.example.com."
- The request message typically includes:
  - The HTTP method (usually GET for fetching resources).
  - The URL of the homepage ("www.example.com").
  - Headers containing additional information such as user-agent, accepted language, accepted encoding, etc.
  - Optionally, if Bob has cookies related to the website, they may be included in the request headers.

Example of a request message:

```
GET / HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36

Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.9

Connection: keep-alive
```

Response Message Preparation by the Server:

- The server hosting "www.example.com" receives the HTTP request message from Bob's browser.
- It processes the request and prepares an HTTP response message to send back to Bob's browser.

- The response message typically includes:
  - The HTTP status code indicating the outcome of the request (e.g., 200 for success, 404 for not found).
  - Headers containing additional information such as content-type, content-length, server information, etc.
  - The actual content of the requested resource (in this case, the homepage HTML).

<u>Example of a response message:</u>

```
HTTP/1.1 200 OK

Date: Sat, 19 Mar 2024 12:00:00 GMT

Server: Apache/2.4.41 (Unix)

Content-Type: text/html; charset=UTF-8

Content-Length: 1234



<!DOCTYPE html>

<html>

<head>

    <title>Example Homepage</title>

</head>

<body>

    <h1>Welcome to Example.com!</h1>

    <!-- Homepage content goes here -->

</body>

</html>
```

In this example:

- The server responds with a status code of 200 OK, indicating that the request was successful.
- The server specifies the content-type as text/html, indicating that the content is HTML.
- The response contains the HTML content of the homepage, which will be rendered by Bob's browser to display the homepage of "www.example.com."

This process demonstrates how the web browser prepares the request message to fetch the homepage, and the server prepares the response message containing the requested homepage content to be displayed in the browser.

6. When Alice wants to access the website "www.example.com" from her web browser, the Domain Name System (DNS) is responsible for translating the human-readable domain name "www.example.com" into the corresponding IP address of the server hosting the website. This translation process involves multiple stages, and both recursive and iterative DNS query approaches can be used to resolve the domain name. Let's explore each approach:

Recursive DNS Query:

A. In a recursive DNS query, Alice's DNS resolver (typically provided by her ISP or configured manually) is responsible for resolving the domain name on her behalf.
B. The stages involved in processing a recursive DNS query are as follows:

   Local DNS Cache Lookup: Alice's DNS resolver first checks its local cache to see if it has a recent record of the IP address for "www.example.com." If the information is found and hasn't expired, the resolver can immediately return the IP address to Alice's browser.

Root DNS Server Query: If the information is not found in the local cache or has expired, Alice's DNS resolver initiates a recursive query starting from the root DNS servers. It sends a query asking for the IP address of the authoritative DNS server responsible for the top-level domain (.com).

Top-Level Domain (TLD) DNS Server Query: The root DNS servers respond to the resolver with the IP address of the TLD DNS server responsible for the ".com" domain. The resolver then queries this TLD DNS server for the IP address of the authoritative DNS server for "example.com."

Authoritative DNS Server Query: The TLD DNS server responds with the IP address of the authoritative DNS server for "example.com." The resolver then sends a query directly to this authoritative DNS server, asking for the IP address of "www.example.com."

IP Address Resolution: The authoritative DNS server for "example.com" responds with the IP address of "www.example.com." This IP address is cached by Alice's DNS resolver for future use, and the resolver returns the IP address to Alice's browser.

Accessing the Website: Alice's web browser now has the IP address of "www.example.com" and can proceed to establish a connection to the server hosting the website to retrieve its content.

Iterative DNS Query:

1.In an iterative DNS query, Alice's DNS resolver performs each stage of the resolution process independently and iteratively. It receives referrals to other DNS servers during the resolution process and continues querying until it obtains the final IP address.

2. The stages involved in processing an iterative DNS query are similar to those in the recursive approach, but instead of resolving the entire query on behalf of Alice, the resolver receives referrals and continues the query process iteratively until it obtains the final IP address.

3. At each stage, the resolver sends queries to DNS servers, receives referrals or IP addresses in response, and continues querying until it reaches the authoritative DNS server for "www.example.com." Once the IP address is obtained, it is returned to Alice's browser for accessing the website.

In summary, both recursive and iterative DNS query approaches involve multiple stages of querying DNS servers to resolve the domain name to its corresponding IP address. However, in a recursive query, the resolver performs the entire resolution process on behalf of the client, while in an iterative query, the resolver performs the resolution process iteratively, receiving referrals and continuing the query until it obtains the final IP address.