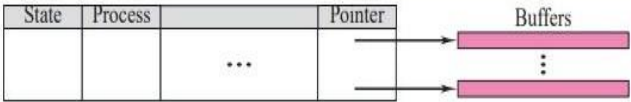
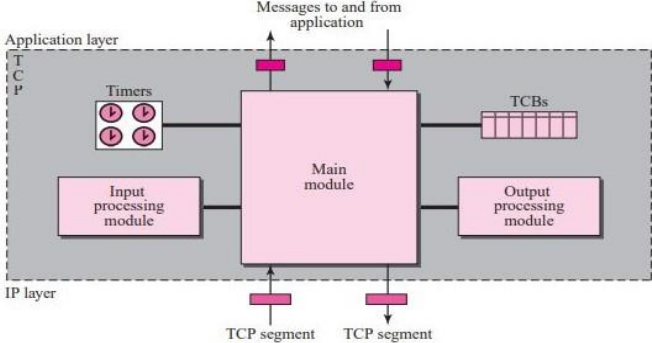


	SET-A
Q. No	Answer with choice variable
1	The remote method invocation _____ a. Allows a process to invoke memory on a remote object b. Allows a thread to invoke a method on a remote object c. Allows a thread to invoke memory on a remote object d. Allows a process to invoke a method on a remote object
2	To differentiate many network services a system supports _____. a. Variables b. Sockets c. Ports d. Service names
3	Which socket type is used for TCP? a. SOCK_DGRAM b. SOCK_STREAM c. SOCK_RAW d. SOCK_TCP
4	What is the one difference between the close() and shutdown() methods for closing a socket? a. close() flushes buffers before closing the socket. b. shutdown() closes the socket immediately. c. close() can indicate which buffers to flush with an integer priority. d. shutdown() does not allow immediate socket closure.
5	A process that is based on IPC mechanism which executes on different systems and can communicate with other processes using message-based communication, is called _____ a. Local Procedure Call b. Inter Process Communication c. Remote Procedure Call d. Remote Machine Invocation
6	A collection of hyperlinked documents on the internet forms: a. World Wide Web (WWW) b. E-mail system c. Mailing list d. Hypertext markup language
7	In the _____ encoding scheme, 24 bits become 4 characters, and eventually are sent as 32 bits. a. 8bit b. 16bit c. base 64 d. binary
8	Simple mail transfer protocol (SMTP) utilizes _____ as the transport layer protocol for electronic mail transfer. a. TCP b. UDP c. HTTP d. SCTP
9	_____ is more powerful and complex than _____ a. POP3; IMAP4 b. IMAP4; POP3 c. SMTP; POP3 d. SMTP; IMAP4
	Which of the following is present in both an HTTP request line and a status line?

10	<p>a. HTTP version number</p> <p>b. URL</p> <p>c. Method</p> <p>d. None of the mentioned</p>
-----------	---

Part – B
(4 x 10 = 40 Marks)

11	<p>You are a network engineer monitoring network traffic on a web server. A user on a remote client initiates a connection to your server to download a large file. Describe how the Processing Module in your server's TCP package handles the data during this file transfer.</p> <p>(Each explanation with diagram 2 marks)</p> <p>1. Transmission Control Blocks (TCBs)</p> <div style="text-align: center;">  </div> <p>2. Timers</p> <p>3. Main Module</p> <p>4. Input Processing Module</p> <p>5. Output Processing Module</p> <div style="text-align: center;">  </div>
-----------	---

OR

12	<p>Explain the following:</p> <p>i. TCP & UDP System Calls with syntax.</p> <p>ii. Socket Descriptor Table (SDT) with a neat diagram</p> <p>i. System calls</p> <ul style="list-style-type: none"> System Calls – An interface between process and operating systems. <p>It provides</p> <ul style="list-style-type: none"> The services of the operating system to the user programs via Application Program Interface(API). An interface to allow user-level processes to request services of the operating system.
-----------	---

- System calls are the only entry points into the kernel system.

TCP Syntax:

```
int socket(int domain, int type, int protocol);
```

```
int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

```
int listen(int sockfd, int backlog);
```

```
int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

```
int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

```
ssize_t send(int sockfd, const void *buf, size_t len, int flags);
```

```
ssize_t recv(int sockfd, void *buf, size_t len, int flags);
```

```
int close(int sockfd);
```

Note: Instead of send & receive students can give the syntax for read & Write.

UDP Syntax:

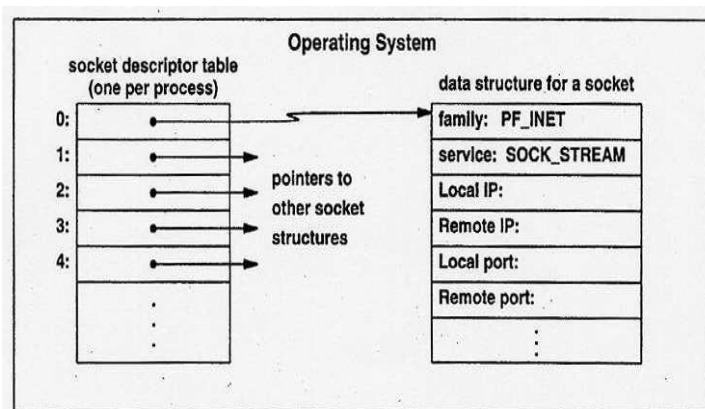
```
ssize_t sendto(int sockfd, const void *buf, size_t len, int flags, const struct sockaddr *dest_addr, socklen_t addrlen);
```

```
ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags, struct sockaddr *src_addr, socklen_t *addrlen);
```

ii. SDT

A socket descriptor table (SDT) is a data structure maintained by the operating system to manage network connections, particularly network-related system calls. Each entry in the table, called a socket descriptor, represents a network communication endpoint and includes information such as socket type (TCP, UDP, etc.), local and remote IP addresses, ports, and connection status. When a program wants to establish a network connection, it uses system calls such as `socket()` to create a socket and obtain a socket descriptor. The socket descriptor is then used in subsequent system calls such as `bind()`, `listen()`, `connect()`, `send()`, `recv()`, and `close()` to manage and communicate over the network.

system calls facilitate interaction between programs and the operating system, and socket descriptor tables help manage network connections by providing a structured way to represent and handle network communication endpoints.



- 13 Create a version of the echo server that will allow connectionless transfer and client code will have to allow exchange of messages with the server. Implement this using the proper mechanism.

Understanding it is transport layer protocol by considering above highlighted points in the text – 2 marks

Figure out scenario – 2 mark

Algorithm/Pseudo code – 2 marks

Program – 4 marks

Protocol is UDP, the following coding may be used for implementing UDP

```
#include<sys/socket.h>
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<netinet/in.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<sys/types.h>
int main(int argc,char *argv[])
{
    int sd;
    char buff[1024];
    struct sockaddr_in cliaddr,servaddr;
    socklen_t clilen;
    clilen=sizeof(cliaddr);

    /*UDP socket is created, an Internet socket address structure is filled with
    wildcard address & server's well known port*/

    sd=socket(AF_INET,SOCK_DGRAM,0);
    if (sd<0)
    {
```

```

        perror ("Cannot open Socket");

        exit(1);

    }

    bzero(&servaddr,sizeof(servaddr));

    /*Socket address structure*/

    servaddr.sin_family=AF_INET;

    servaddr.sin_addr.s_addr=htonl(INADDR_ANY);

    servaddr.sin_port=htons(5669);


    /*Bind function assigns a local protocol address to the socket*/

    if(bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)

    {

        perror("error in binding the port");

        exit(1);

    }

    printf("%s", "Server is Running...\n");

    while(1)

    {

        bzero(&buff,sizeof(buff));


        /*Read the message from the client*/

        if(recvfrom(sd,buff,sizeof(buff),0,(struct sockaddr*)&cliaddr,&clilen)<0)

        {

            perror("Cannot rec data");

            exit(1);

        }

        printf("Message is received \n",buff);


        /*Sendto function is used to echo the message from server to client side*/

        if(sendto(sd,buff,sizeof(buff),0,(struct sockaddr*)&cliaddr,clilen)<0)

```

```

        {

                perror("Cannot send data to client");

                exit(1);

        }

        printf("Send data to UDP Client: %s",buff);

    }

    cloSe(sd);

    return 0;

}

```

Client: udpclient.c

```

#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<netinet/in.h>
#include<netdb.h>

```

```

int main(int argc,char*argv[])
{

```

```

    int sd;

    char buff[1024];

    struct sockaddr_in servaddr;

    socklen_t len;

    len=sizeof(servaddr);

```

/*UDP socket is created, an Internet socket address structure is filled with wildcard address & server's well known port*/

```

    sd = socket(AF_INET,SOCK_DGRAM,0);

```

```

if(sd<0)

{

    perror("Cannot open socket");

    exit(1);

}

bzero(&servaddr,len);


/*Socket address structure*/

servaddr.sin_family=AF_INET;

servaddr.sin_addr.s_addr=htonl(INADDR_ANY);

servaddr.sin_port=htons(5669);

while(1)

{

    printf("Enter Input data : \n");

    bzero(buff,sizeof(buff));


    /*Reads the message from standard input*/

    fgets(buff,sizeof (buff),stdin);


    /*sendto is used to transmit the request message to the server*/

    if(sendto (sd,buff,sizeof (buff),0,(struct sockaddr*)&servaddr,len)<0)

    {

        perror("Cannot send data");

        exit(1);

    }

    printf("Data sent to UDP Server:%s",buff);

    bzero(buff,sizeof(buff));


    /*Receiving the echoed message from server*/

    if(recvfrom (sd,buff,sizeof(buff),0,(struct sockaddr*)&servaddr,&len)<0)

    {

```

	<pre> perror("Cannot receive data"); exit(1); } printf("Received Data from server: %s",buff); } close(sd); return 0; } </pre>
--	--

OR

14

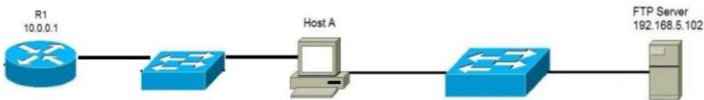
Compare the TCP header and the UDP header. List the fields in the TCP header that are missing from UDP header. Give the reasons for their absence.

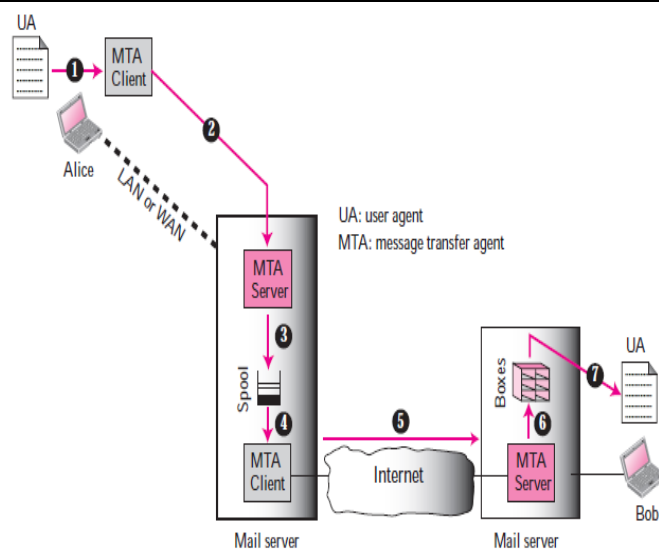
Basis	Transmission Control Protocol (TCP)	User Datagram Protocol (UDP)
Type of Service	TCP is a connection-oriented protocol. Connection orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data.	UDP is the Datagram-oriented protocol. This is because there is no overhead for opening a connection, maintaining a connection, or terminating a connection. UDP is efficient for broadcast and multicast types of network transmission.
Reliability	TCP is reliable as it guarantees the delivery of data to the destination router.	The delivery of data to the destination cannot be guaranteed in UDP.
Error checking mechanism	TCP provides extensive error-checking mechanisms. It is because it provides flow control and acknowledgment of data.	UDP has only the basic error-checking mechanism using checksums.
Acknowledgment	An acknowledgment segment is present.	No acknowledgment segment.

		Sequence	Sequencing of data is a feature of Transmission Control Protocol (TCP). this means that packets arrive in order at the receiver.	There is no sequencing of data in UDP. If the order is required, it has to be managed by the application layer.
		Speed	TCP is comparatively slower than UDP.	UDP is faster, simpler, and more efficient than TCP.
		Retransmission	Retransmission of lost packets is possible in TCP, but not in UDP.	There is no retransmission of lost packets in the User Datagram Protocol (UDP).
		Header Length	TCP has a (20-60) bytes variable length header.	UDP has an 8 bytes fixed-length header.
		Weight	TCP is heavy-weight.	UDP is lightweight.
		Handshaking Techniques	Uses handshakes such as SYN, ACK, SYN-ACK	It's a connectionless protocol i.e. No handshake
		Broadcasting	TCP doesn't support Broadcasting.	UDP supports Broadcasting.
		Protocols	TCP is used by HTTP, HTTPs, FTP, SMTP and Telnet.	UDP is used by DNS, DHCP, TFTP, SNMP, RIP, and VoIP.
		Stream Type	The TCP connection is a byte stream.	UDP connection is a message stream.
		Overhead	Low but higher than UDP.	Very low.
		Applications	This protocol is primarily utilized in situations when a safe and trustworthy communication procedure is necessary, such as in email, on the web surfing, and in military services.	This protocol is used in situations where quick communication is necessary but where dependability is not a concern, such as VoIP, game streaming, video, and music streaming, etc.

The fields that are missing in the UDP header but present in the TCP header are - The sequence number, acknowledge number, and Window fields.

Reason for absence
UDP is designed to be a lightweight, connectionless protocol. It is used in situations where low overhead is more critical than guaranteed delivery or extensive error checking. Because of its simplicity, UDP lacks features like sequencing, acknowledgments, and complex control mechanisms found in TCP. This simplicity makes UDP faster and more efficient for certain applications (such as real-time multimedia streaming and online gaming) where the timely delivery

	<p>of data is more important than ensuring that all packets are received in order. However, this lack of features also means that UDP does not provide the same level of reliability and error recovery as TCP. Applications using UDP must handle issues like packet loss and sequencing at a higher layer if necessary</p>
15	<p>i) Telnet uses TCP port 23. Does this imply that the telnet connection initiated by a user connecting to the router from a computer uses TCP port 23.</p> <p>Ans: 5m</p> <p>This is not true. When the user initiates a connection to the router, the TCP header in the connection would have a source port and destination port number. The destination port number would be port 23, indicating that the request is being initiated to the telnet server service which is identified by port 23. The source would be a random port number assigned by the TCP/IP on the operating system of the computer.</p> <p>ii) Zen accesses the host A machine to download the ASCII file named “Zen_file” in compressed form from the FTP Server. The file resides in the path “ftpd/user/Zen”. Identify the suitable protocol and suggests Zen in framing the appropriate commands to download the file.</p>  <p>Answer:- 5m</p> <p>File Transfer Protocol</p> <pre> 220 (Service ready) USER Zen 331 (User name OK. Password?) PASS yyy 230 (User login OK) PORT 1267 150 (Data Connection opens shortly) TYPE ASCII 200 (OK) STRU F 200 (OK) MODE C 200 (OK) RETR ftpd/user/Zen/Zen_file 250 (OK) (Data Transfer from server to client) 226 (Closing data connection) QUIT 221 (Service closing) </pre>
OR	
16	<p>i. In Email communication system, a sender is connected to the mail server via LAN/WAN, identify the component requirements and draw the system architecture.</p> <p>ii. Compare POP3 & IMAP 4</p> <p>i. Identification of components & System architecture (5m)</p>



ii.

POP3(Post Office Protocol, version 3):

- The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the mail server. Mail access starts with the client when the user needs to download its e-mail from the mailbox.
- The client opens a connection to the server on TCP port. It then sends its user name and password to access the mailbox. The user can then list and retrieve the mail messages, one by one.

IMAP4(Internet Mail Access Protocol, version 4): It is more powerful and more complex.

IMAP4 provides the following extra functions:

- A user can check the e-mail header prior to downloading.
- A user can search the contents of the e-mail for a specific string of characters prior to downloading.
- A user can partially download e-mail. This is especially useful if bandwidth is limited and the e-mail contains multimedia with high bandwidth requirements.
- A user can create, delete, or rename mailboxes on the mail server.

A user can create a hierarchy of mailboxes in a folder for e-mail storage.

- 17**
- In SMTP, show the connection establishment phase from aaa@xxx.com to bbb@yyy.com. (5)
 - In SMTP, show the message transfer phase from aaa@xxx.com to bbb@yyy.com. The message is "Good morning my friend." (2.5)
 - In SMTP, show the connection termination phase from aaa@xxx.com to bbb@yyy.com. (2.5)
- i) Connection Establishment Phase (5)
- DNS Resolution: The sender's SMTP client performs a DNS query to resolve the MX (Mail Exchange) record for the domain yyy.com. This provides the address of the mail server responsible for handling incoming mail for yyy.com.
- TCP Connection: The SMTP client initiates a TCP connection to the SMTP server obtained from

the MX record at port 25, the default port for SMTP.

SMTP Greeting: The client sends a "HELO" or "EHLO" command to the server to introduce itself and initiate the SMTP session.

Server Response: The server responds with a message acknowledging the client's greeting and indicating its readiness to proceed with the SMTP transaction.

ii) Message Transfer Phase (2.5)

Sender Identification: The SMTP client initiates the email transfer by issuing the "MAIL FROM: aaa@xxx.com" command, identifying the sender.

Recipient Identification: The client then issues the "RCPT TO: bbb@yyy.com" command, specifying the recipient.

Message Data: The client sends the email content using the "DATA" command and transmits the message content, in this case, "Good morning my friend."

Termination and Confirmation: After the message is transmitted, the client sends a period (.) on a line by itself to indicate the end of the message.

iii) Connection Termination Phase (2.5)

Closing the Connection: The client issues the "QUIT" command to initiate the termination of the SMTP session.

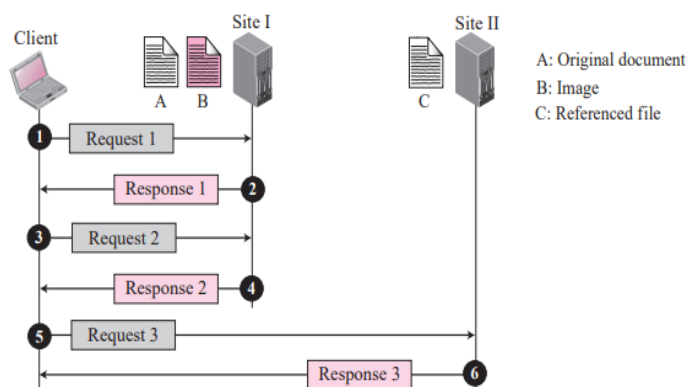
Server Acknowledgment: The server responds with a confirmation message, acknowledging the termination request.

Connection Closed: The client and server close the TCP connection, ending the SMTP communication.

OR

- 18** Assume you need to retrieve a document that contains one reference to another text file and one reference to a large image. The main document and the image are stored in two separate files on the same site (file A and file B); the referenced text file is stored on another site (file C). Demonstrate the three transactions to see the whole document. Also, give the uniform resource locator format to locate any kind of information on the Internet.

Demonstration of three transactions: (5m)



ii. uniform resource locator format with explanation(5m)

	<div>Protocol</div> <div>:</div> <div>//</div> <div>Host</div> <div>:</div> <div>Port</div> <div>/</div> <div>Path</div>
--	--