Q. No	**SET-C** **Answer with choice variable**
1	The Transport layer is responsible for which type of communication a. node to node b. host to host c. **process to process** d. end to end
2	TCP allows the sending process to transmit data in _____________bytes and the receiving process to receive data in _____________ bytes. a. message, message b. **stream, stream** c. block, block d. fragment, fragment
3	What are the two modes of operation for a listening server process? a. Telnet and HTTP b. Telnet and FTP c. **Iterative and concurrent** d. TCP and concurrent
4	Which of the following information is typically stored in a TCB? a. IP address of the router b. **Port numbers, sequence numbers, and acknowledgment numbers** c. MAC address of the local host d. Subnet mask
5	In the listen system call of a socket, which parameter represents the maximum number of connections allowed on the incoming queue at any given time? a. ConnectLimit b. QueueSize c. **Backlog** d. ConnectionCount
6	An inverted hierarchical tree structure with one root node at top and a maximum of DNS can be pictured as a**) 128 Levels** b) 129 Levels c) 130 Levels d) 131 Levels e) 132 Levels
7	An email client needs to know the _________ of its initial SMTP server. a**) IP address** b) MAC address c) URL d) Name

8	Which email protocol is primarily used for downloading emails from the server to a local device and stores them locally, potentially leading to inconsistent email access across multiple devices? **a) POP3** b) IMAP c) MIME d) SMTP
9	Identify the incorrect HTTP status codes a) 200 OK b) 400 Bad Request c) 301 Moved permanently d) **304 Not Found**
10	Which of the following is true about POP3? a) It supports synchronization of emails across devices b) **It keeps emails on the server even after they are retrieved** c) It encrypts emails during transmission d) It is a web-based email protocol

Part – B
(4 x 10 = 40 Marks)

11 Design a server program that provides real-time date and time information to multiple clients over a network. The server should be able to handle numerous client connections without blocking or slowing down, ensuring a smooth user experience. 10

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>


#define SERVER_PORT 12345

#define BUFFER_SIZE 1024


void handle_client(int client_socket) {

    // Receive request from the client
    char buffer[BUFFER_SIZE];
    int bytes_received = recv(client_socket, buffer, BUFFER_SIZE, 0);
    if (bytes_received < 0) {
        perror("recv failed");
        close(client_socket);
        return;
    }

    // Get current date and time
    time_t current_time;
    time(&current_time);
    struct tm *time_info = localtime(&current_time);

    // Format date and time string
    char time_string[100];
    strftime(time_string, sizeof(time_string), "%c", time_info);

    // Send date and time to the client
    send(client_socket, time_string, strlen(time_string), 0);

    close(client_socket);
}

int main() {
    // Create TCP socket
    int server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (server_socket < 0) {
        perror("socket creation failed");
        exit(EXIT_FAILURE);
    }
    // Define server address
    struct sockaddr_in server_addr;
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(SERVER_PORT);

    // Bind socket to the server address
    if (bind(server_socket, (struct sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    // Listen for incoming connections
    if (listen(server_socket, 5) < 0) {
        perror("listen failed");
        exit(EXIT_FAILURE);
    }

    printf("Server listening on port %d\n", SERVER_PORT);

    while (1) {
        // Accept incoming connection
        struct sockaddr_in client_addr;
        socklen_t client_addr_len = sizeof(client_addr);
        int client_socket = accept(server_socket, (struct sockaddr*)&client_addr,
&client_addr_len);
        if (client_socket < 0) {
            perror("accept failed");
            continue;
        }


        // Create a child process to handle the client
        pid_t pid = fork();
        if (pid == 0) {
            close(server_socket); // Child process doesn't need the server socket
            handle_client(client_socket);
            exit(EXIT_SUCCESS);
        }
        else if (pid > 0) {
            close(client_socket); // Parent process doesn't need the client socket
        }
        else {
            perror("fork failed");
        }
    }


    close(server_socket);
    return 0;
}
```

12 Write programs for Network Echo Server "Echo_s", an Echo Client "Echo_c", and a Log Server "Log_s" that uses unreliable transport protocol. 10

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 sockadddr*)&cliaddr,clilen)<0)
  {
perror("Cannot send data to client");
exit(1);
      }
```

```
            printf("Send data to UDP Client: %s",buff);
}
cloSe(sd);
return 0;
}
```

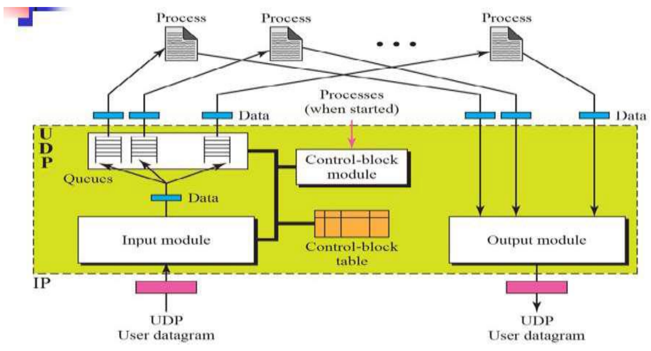
Client: udpclient.c

```
#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)
            {
                        perror("Cannot receive data");
                       exit(1);
            }
printf("Received Data from server: %s",buff);
  }
close(sd);
return 0;
}
```

13 Enumerate all the modules of UDP package and explain Input output module with neat sketch **10**

Five components

- Control-block table
- Input queues
- Control-block module
- Input module
- Output module



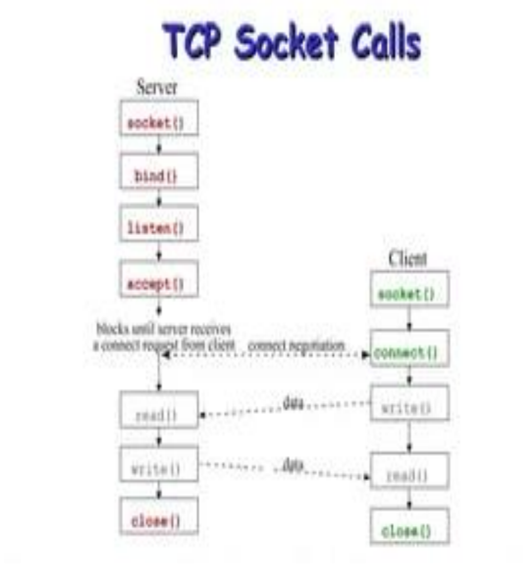
Input module

It receives a user datagram from IP.

●It searches the control block table to find an entry having the same port number as this user datagram
If found -> the module uses the info in the entry to enqueue the data.

	□If not found -> it generates an ICMP message.
	
	1.UDP_INPUT_Module(user_datagram)
	2.{
	3.Look for the entry in the control_blocktable
	4.if (found)
	5.{
	6.Check to see if a queue is allocated
	7.Allocate a queue
	8.else
	9.Enqueuethe data
	10.} //end if
	11.else
	12.{
	13.Ask ICMP to send an “unreachable port” message
	14.Discard the user datagram
	15.}//end else
	16.Return
	17.} //end module
	Output module
	
	Itis responsible for creating and sending user datagrams
	1.UDP_OUTPUT_MODULE(Data)
	2.{
	3.Create a user datagram
	4.Send the user datagram
	5.Return.
	6.}	

14	Evaluate and synthesize a comprehensive diagram illustrating the precise sequence of system calls essential for the implementation of both TCP clients and servers. Subsequently, analyze and justify the rationale behind each chosen system call in the context of TCP network programming.	10



1. TCP Server:

Socket Creation (socket): The server creates a socket using the socket() system call. This socket will be used for listening to incoming connections.

Binding (bind): The server binds the socket to a specific IP address and port number using the bind() system call. This step associates the server with a specific network interface and port.

Listening (listen): The server sets the socket to the listening state using the listen() system call. This allows the server to accept incoming connection requests from clients.

Accepting Connections (accept): The server uses the accept() system call in a loop to accept incoming connections. When a client connects, this call returns a new socket dedicated to that client's communication.

Data Exchange (send and recv): The server communicates with the client(s) using the send() and recv() system calls to send and receive data over the established connections.

Closing Connections (close): After communication is complete, the server uses the close() system call to close the individual client sockets and release resources.

Terminating (close): When the server is done, it uses the close() system call to close the main listening socket and terminate the server.

TCP Client:

	Socket Creation (socket): The client creates a socket using the socket() system call. This socket will be used for communication with the server. Connecting to Server (connect): The client uses the connect() system call to establish a connection to the server by specifying the server's IP address and port number. Data Exchange (send and recv): The client communicates with the server using the send() and recv() system calls to send and receive data over the established connection. Closing Connection (close): After communication is complete, the client uses the close() system call to close the socket and release resources.	

15	Bob is an internet user who wants to visit a website by typing its domain name in his web browser. Can you describe the functioning of DNS, its significance for Bob's internet access, and elaborate on the process of DNS resolution and the hierarchical structure of DNS within the internet ecosystem? The DNS resolution process: Bob enters a domain name (e.g., www.example.com) in his web browser. The browser contacts a DNS resolver (often provided by Bob's ISP). The DNS resolver checks its cache for the IP address associated with the domain. If found, it returns the IP. If not in the cache, the resolver initiates a query to the root DNS server. The root server directs the resolver to the top-level domain (TLD) server (e.g., .com). The TLD server directs the resolver to the authoritative DNS server for the specific domain (e.g., example.com). The authoritative server returns the IP address for www.example.com to the resolver. The resolver stores the IP in its cache and returns it to Bob's browser. Bob's browser uses the IP to establish a connection and load the website. The hierarchical structure of DNS: Root DNS Servers: At the top of the hierarchy, they direct queries to TLD servers.	10

	Top-Level Domain (TLD) Servers: Manage domain extensions (e.g., .com, .org). Authoritative DNS Servers: Hold DNS records for specific domains. DNS Resolvers: Used by ISPs or configured on individual devices to facilitate DNS queries. This hierarchical system efficiently resolves domain names to IP addresses, ensuring users can access websites with ease.	
16	Alice is setting up a small office network with several computers and devices. She wants to use DHCP to automatically assign IP addresses to the devices. Explain the concept of DHCP, its significance for Alice's network, how DHCP operates within her network, and outline the steps Alice should follow to configure DHCP effectively in her network. Provide a detailed response covering these aspects. DHCP (Dynamic Host Configuration Protocol): Concept: DHCP is a network protocol that automatically assigns IP addresses and network configuration settings to devices on a network, eliminating the need for manual configuration. Significance: It simplifies network management, reduces IP conflicts, and ensures efficient resource allocation. How DHCP Operates in Alice's Network: Request: When a device connects to Alice's network, it sends a DHCP request for an IP address to the DHCP server. Offer: The DHCP server responds with an IP address offer, along with other network configuration details, like subnet mask and gateway. Request Acknowledgment: The device sends a formal request for the offered IP address. Acknowledgment: The DHCP server acknowledges the request and finalizes the IP address assignment. Steps to Configure DHCP Effectively: Install DHCP Server: Ensure a DHCP server is set up within the network. This can be a dedicated server or a router with DHCP capabilities. IP Range Selection: Define a range of IP addresses that the DHCP server can allocate to devices. Be sure to consider the number of devices in the network. Lease Duration: Set a lease duration for IP addresses. This determines how long a device can use an assigned IP before it must renew the lease.	10

	Subnet Configuration: Ensure the subnet mask and gateway settings are correctly configured in the DHCP server. DNS Configuration: Specify DNS server addresses for the devices, allowing them to resolve domain names. Static IP Assignments: If certain devices require static (fixed) IP addresses, configure these reservations in the DHCP server. Testing and Monitoring: Test the DHCP setup to ensure devices receive IP addresses. Monitor the DHCP server for any issues or address conflicts. Configuring DHCP effectively simplifies network management and ensures devices can easily connect to Alice's office network.	

17	Alice is a web developer, and she's working on a new website for her client. She's using HTTP to communicate between the web browser and the web server. Discuss HTTP, its importance in Alice's web development project, the process when a user enters a website's URL in their browser's address bar, and the key components of both HTTP requests and HTTP replies. HTTP (Hypertext Transfer Protocol): Importance: Vital for web development, it enables data exchange between browsers and servers. Process when a user enters a website's URL: URL Entry: User inputs the URL. DNS Resolution: Browser resolves domain to IP. TCP Connection: Browser connects to the server via TCP. HTTP Request: Browser sends a request (method, URL, headers, optional body). Server Processing: Server processes the request, prepares a response. HTTP Reply: Server sends a reply (status code, headers, body). Rendering: Browser interprets and displays the webpage. Key Components: HTTP Requests: Method, URL, Headers, Body (optional). HTTP Replies: Status Code, Headers, Body. HTTP facilitates web development by ensuring seamless data exchange between browsers and servers. Understanding its components is essential for effective development.	10

18	Alice, one of Bob's employees, frequently works remotely from different locations. She uses multiple devices, including her laptop, tablet, and smartphone, to access her work emails. Alice needs a way to manage her emails and folders consistently across all devices, ensuring that her email actions (read/unread, delete, move) are synchronized. Justify and explain about the email protocol for Alice to employ and to ensure consistent email synchronization across all her devices? (10M) Alice should use **IMAP (Internet Message Access Protocol)** for consistent email synchronization across all her devices. IMAP keeps emails and folders synchronized in real-time, allowing her to manage her emails consistently from any device. To set up IMAP for her email, Alice should follow these general steps: Open her email client on each device (e.g., laptop, tablet, smartphone). Add a new email account, specifying the email address and password. Choose IMAP as the account type. Configure the incoming and outgoing mail server settings provided by her email service provider. By using IMAP, Alice can effectively manage her email and folders consistently across all her devices, ensuring that her email actions, such as read/unread, delete, and move, are synchronized and up-to-date.	10