Dr. M.G.R.
EDUCATIONAL AND RESEARCH INSTITUTE
DEEMED TO BE UNIVERSITY
University with Graded Autonomy Status
(An ISO 21001 : 2018 Certified Institution)
Periyar E.V.R. High Road, Maduravoyal, Chennai-95. Tamilnadu, India.

RECORD NOTEBOOK

EBCS22L05 – NETWORK PROGRAMMING LAB

DEPARTMENT

OF

COMPUTER SCIENCE AND ENGINEERING

NAME : KUMARAN T

REGISTER NO : 231061201005

COURSE : B. TECH CSE (P.T)

YEAR/SEM/SEC : II/III

2024-2025 (ODD SEMESTER)

Dr. M.G.R.
EDUCATIONAL AND RESEARCH INSTITUTE
DEEMED TO BE UNIVERSITY
University with Graded Autonomy Status
(An ISO 21001 : 2018 Certified Institution)
Periyar E.V.R. High Road, Maduravoyal, Chennai-95. Tamilnadu, India.

BONAFIDE CERTIFICATE

REGISTER NO: 231061201005

NAME OF LAB: NETWORK PROGRAMMING LAB - (EBCS22L05)

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

Certified that, this Record note book is a bonafide record of work done by KUMARAN T of **II Year B. Tech / CSE** in the **NETWORK PROGRAMMING LAB (EBCS22L05)** during the year **2024 -2025 .**

Signature of Lab-in-Charge **Signature of Head of Dept**

Submitted for the Practical Examination held on ----------

Internal Examiner **External Examiner**

INDEX

Exp.No	Date	Name of the Experiment	Page No.	Staff Signature
1.		Networking Commands with options		
2.		Socket Program to extend communication between two deferent ends using TCP		
3.		Socket program to extend communication between two deferent ends using UDP		
4.		Create a Socket (TCP) between two computers and enable file transfer between them		
5.		Design a TCP concurrent server to echo given set of sentences using poll functions		
6.		Implement Concurrent Time Server application using UDP to execute the program at remote server. Client sends a time request to the server sends its system time back to the client. Client displays the result		
7.		Implementation of RPC in server-client model		
8.		Implementation of ARP/RARP		
9.		HTTP Socket program to download a web page		
10.		File transfer in Client-Server architecture using following methods A)Using RS232C b)Using TCP/IP		
11.		To implement RMI (Remote Method Invocation)		
12.		Write a network program to broadcast/multicast a message to a group in the same network		
13.		Demonstration of Network Simulators		

Ex. No: 1 **Date:**

NETWORKING COMMANDS WITH OPTIONS

AIM

To work on networking commands with options in Windows Operating System.

ALGORITHM

Step1. Start the program

Step2. Open the command prompt

Step3. Enter the commands along with the proper options

Step4. View the Command Output

PROGRAM

1. **ping**

   This is used to provide a basic connectivity test between the requesting host and the destination host.

2. **ipconfig**

   Displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings.

3. **tracert**

   This command is available only if the Internet Protocol (TCP/IP) protocol is installed as a component in the properties of a network adapter in Network Connections.

4. **nslookup**

   nslookup is the name of a program that lets an Internet server administrator or any computer user enter a host name (for example, "whatis.com") and find out the corresponding IP address or domain name system (DNS) record. The user can also enter a command for it to do a reverse DNS lookup and find the host name for an IP address that is specified.

5. **ipconfig/all**

6. Without the use of any parameter, the command shows only the basic network information. But information about the DNS and DHCP servers is not displayed by default. To show all the information about your network adapter, you will need to use the /all parameter.

7. **getmac**

   The getmac (short for get MAC address) is a simple Windows network command-line utility used to find the physical address of the network adapters (NIC) in a computer. This tool is typically used in troubleshooting network issues.

8. **net**

   The following Net Commands can be used to perform operations on Groups, users, account policies, shares, and so on.

9. **arp**

   arp (**Address Resolution Protocol**) Maps IP address to MAC (Media Access Control) / physical address? nslookup. Maps domain name to IP address.

OUTPUT

```
Select Command Prompt
Microsoft Windows [Version 10.0.22621.674]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mk846>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Unknown adapter Local Area Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet 3:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::5e39:6e80:1e23:389e%16
   IPv4 Address. . . . . . . . . . . : 192.168.192.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::2b5f:a395:cbb9:8dd3%17
   IPv4 Address. . . . . . . . . . . : 172.16.184.122
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : fe80::1e5f:2bff:fe8e:1744%17
                                       172.16.1.1

C:\Users\mk846>ipconfig/all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : Manish
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Ethernet 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : ExpressVPN TAP Adapter
   Physical Address. . . . . . . . . : 00-FF-41-6B-8D-6E
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes

Unknown adapter Local Area Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : ExpressVPN Wintun Driver
   Physical Address. . . . . . . . . :
   DHCP Enabled. . . . . . . . . . . : No
```

```
Ethernet adapter Ethernet 3:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : VirtualBox Host-Only Ethernet Adapter
   Physical Address. . . . . . . . . : 0A-00-27-00-00-10
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::5e39:6e80:1e23:389e%16(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.192.1(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
   DHCPv6 IAID . . . . . . . . . . . : 805961767
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2A-C7-92-AF-A8-64-F1-AF-BD-15
   NetBIOS over Tcpip. . . . . . . . : Enabled

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Microsoft Wi-Fi Direct Virtual Adapter
   Physical Address. . . . . . . . . : A8-64-F1-AF-BD-16
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
```

Select Command Prompt

```
Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #2
   Physical Address. . . . . . . . . : AA-64-F1-AF-BD-15
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) Wi-Fi 6 AX201 160MHz
   Physical Address. . . . . . . . . : A8-64-F1-AF-BD-15
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::2b5f:a395:cbb9:8dd3%17(Preferred)
   IPv4 Address. . . . . . . . . . . : 172.16.184.122(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Lease Obtained. . . . . . . . . . : 31 October 2022 14:16:58
   Lease Expires . . . . . . . . . . : 01 November 2022 11:53:47
   Default Gateway . . . . . . . . . : fe80::1e5f:2bff:fe8e:1744%17
                                       172.16.1.1
   DHCP Server . . . . . . . . . . . : 172.16.1.1
   DHCPv6 IAID . . . . . . . . . . . : 212362481
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2A-C7-92-AF-A8-64-F1-AF-BD-15
   DNS Servers . . . . . . . . . . . : 172.16.1.1
   NetBIOS over Tcpip. . . . . . . . : Enabled

C:\Users\mk846>nslookup google.com
Server:  UnKnown
Address:  172.16.1.1

Non-authoritative answer:
Name:    google.com
Addresses:  2404:6800:4007:806::200e
          142.250.71.14


C:\Users\mk846>tracert facebook.com

Tracing route to facebook.com [157.240.23.35]
over a maximum of 30 hops:

  1     1 ms     1 ms     1 ms  172.16.1.1
  2    20 ms    14 ms    14 ms  103.60.137.65
  3    18 ms    14 ms    16 ms  103.60.138.5
  4    19 ms    13 ms    12 ms  as32934.maa.extreme-ix.net [45.120.251.139]
  5    33 ms    28 ms    17 ms  po102.psw04.maa2.tfbnw.net [129.134.34.157]
  6    14 ms    17 ms    16 ms  173.252.67.161
  7    15 ms    18 ms    14 ms  edge-star-mini-shv-01-maa2.facebook.com [157.240.23.35]

Trace complete.

C:\Users\mk846>ping 172.16.1.1

Pinging 172.16.1.1 with 32 bytes of data:
Reply from 172.16.1.1: bytes=32 time=2ms TTL=64
Reply from 172.16.1.1: bytes=32 time=2ms TTL=64
Reply from 172.16.1.1: bytes=32 time=1ms TTL=64
Reply from 172.16.1.1: bytes=32 time=2ms TTL=64

Ping statistics for 172.16.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 2ms, Average = 1ms
```

```
C:\Users\mk846>getmac

Physical Address    Transport Name
=================== ============================================================
00-FF-41-6B-8D-6E   Media disconnected
N/A                 Media disconnected
A8-64-F1-AF-BD-15   \Device\Tcpip_{BEC9492F-A667-480B-8ACF-ADDFAFA81D5F}
0A-00-27-00-00-10   \Device\Tcpip_{BBC6A28B-453E-4C8C-8C46-E868DFE0EF98}

C:\Users\mk846>net
The syntax of this command is:

NET
    [ ACCOUNTS | COMPUTER | CONFIG | CONTINUE | FILE | GROUP | HELP |
      HELPMSG | LOCALGROUP | PAUSE | SESSION | SHARE | START |
      STATISTICS | STOP | TIME | USE | USER | VIEW ]

C:\Users\mk846>net statistics
Statistics are available for the following running services:

   Workstation

The command completed successfully.


C:\Users\mk846>arp -a

Interface: 192.168.192.1 --- 0x10
  Internet Address      Physical Address      Type
  192.168.192.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  230.0.0.1             01-00-5e-00-00-01     static
Select Command Prompt
Interface: 172.16.184.122 --- 0x11
  Internet Address      Physical Address      Type
  172.16.1.1            00-e2-69-0d-9a-f8     dynamic
  172.16.100.224        b4-8c-9d-ac-b9-57     dynamic
  172.16.101.190        2a-52-a7-69-13-86     dynamic
  172.16.255.255        ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  230.0.0.1             01-00-5e-00-00-01     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

RESULT

Thus, the Network Commands with options for Windows Operating System has been executed successful.

Ex. No: 2 **Date:**

SOCKET PROGRAM TO EXTENT COMMUNICATION BETWEEN TWO DIFFERENT ENDS USING TCP

AIM

To write a java program for implementing Socket program to extent communication between two deferent ends using TCP.

ALGORITHM

SERVER

Step1. Start the program

Step2. Create an unnamed socket for the server using the parameters AF_INET as domain and the SOCK_STREAM as type.

Step3. Name the socket using bind( ) system call with the parameters server_sockfd and the server address (sin_addr and sin_sport).

Step4. Create a connection queue and wait for clients using the listen( ) system call with the number of clients request as parameters.

Step5. Accept the connection using accept( ) system call when client requests for connection.

Step6. Get the message which has to be sent to the client and check that it is not equal to 'Bye'.

Step7. If the message is not equal to 'Bye' then write the message to the client and Goto step 6.

Step8. If the message is 'Bye' then terminate the Process.

Step9. Stop the program

CLIENT

Step1. Start the program

Step2. Create an unnamed socket for client using socket( ) system.

Step3. Call with parameters AF_INET as domain and SOCK_STREAM as type.

Step4. Name the socket using bind( ) system call.

Step5. Now connect the socket to server using connect( ) system call.

Step6. Read the message from the server socket and compare it with 'Bye'.

Step7. If the message is not equal to 'Bye' then print the message to the server output device and repeat the steps 6 & 7.

Step8. Get the message from the client side.

Step9. Write the message to server sockfd and goto step 4.

Step10. If the message is equal to 'Bye'then print good bye message and terminate the process.

Step11. Stop the program

PROGRAM

SERVER

```
import java.net.*;
import java.io.*;
public class Server
{
        //initialize socket and input stream
        private Socket socket = null;
        private ServerSocket server = null;
        private DataInputStream in = null;
        public Server(int port)
        {
                try
                {
                        server = new ServerSocket(port);
                        System.out.println("Server started");
                        System.out.println("Waiting for a client ...");
                        socket = server.accept();
                        System.out.println("Client accepted");
                        // takes input from the client socket
                        in = new DataInputStream(new
BufferedInputStream(socket.getInputStream()));
                        String line = "";
                        // reads message from client until "Over" is sent
                        while (!line.equals("Over"))
```

```
                {
                        try
                        {
                                line = in.readUTF();
                                System.out.println(line);
                        }
                        catch(IOException i)
                        {
                                System.out.println(i);
                        }
                }
                System.out.println("Closing connection");
                socket.close();
                in.close();
        }
        catch(IOException i)
        {
                System.out.println(i);
        }
    }
    public static void main(String args[])
    {
            System.out.println("This program is done by: MANISH SINGH SURYAVANSHI \t 201061101118\n");
            Server server = new Server(5000);
    }
}
```

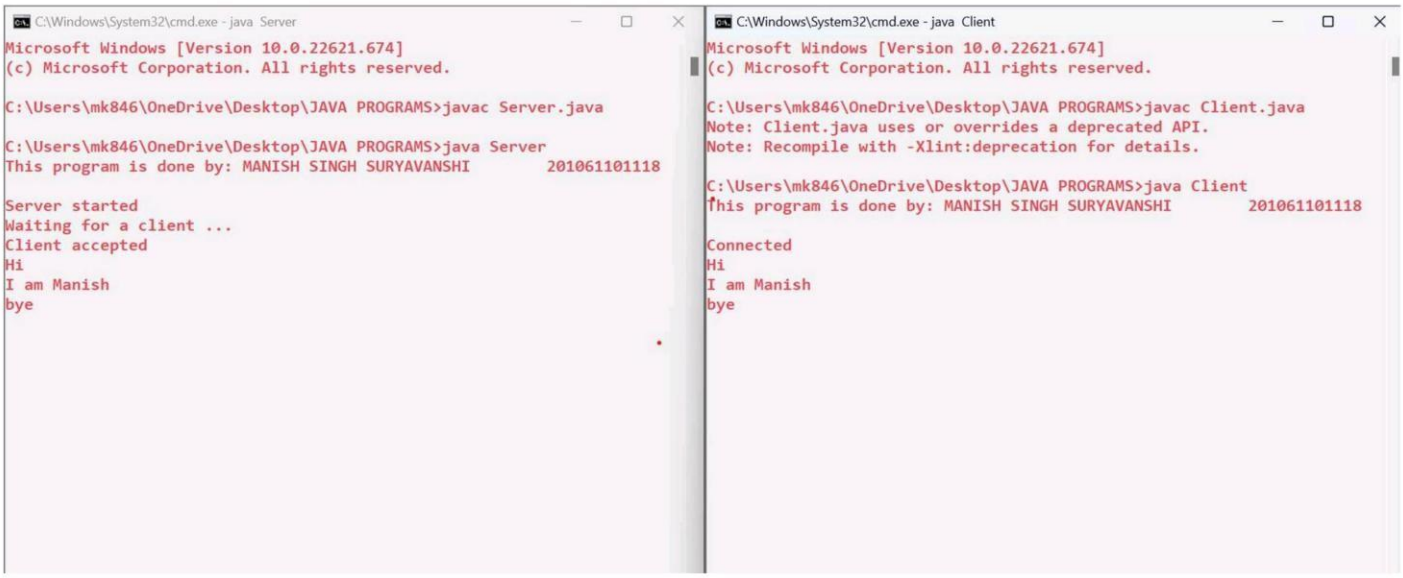
CLIENT

```
import java.net.*;
import java.io.*;
public class Client
{
	private Socket socket = null;
	private DataInputStream input = null;
	private DataOutputStream out = null;
	public Client(String address, int port)
	{
		try
		{
			socket = new Socket(address, port);
			System.out.println("Connected");
			input = new DataInputStream(System.in);
			out = new DataOutputStream(socket.getOutputStream());
		}
		catch(UnknownHostException u)
		{
			System.out.println(u);
		}
		catch(IOException i)
		{
			System.out.println(i);
		}
		String line = "";
```

```
            while (!line.equals("Over"))
            {
                try
                {
                    line = input.readLine();
                    out.writeUTF(line);
                }
                catch(IOException i)
                {
                    System.out.println(i);
                }
            }
            try
            {
                input.close();
                out.close();
                socket.close();
            }
            catch(IOException i)
            {
                System.out.println(i);
            }
        }
        public static void main(String args[])
        {
            System.out.println("This program is done by: MANISH SINGH SURYAVANSHI \t
201061101118\n");
```

```
            Client client = new Client("127.0.0.1", 5000);
        }
    }
```

OUTPUT



RESULT

Thus, the java program for implementing Socket program to extent communication between two deferent ends using TCP is executed and the output is verified successfully

Ex. No: 3 **Date:**

SOCKET PROGRAM TO EXTENT COMMUNICATION BETWEEN TWO DIFFERENT ENDS USING UDP

AIM

To write a java program for implementing Socket program to extent communication between two deferent ends using UDP.

ALGORITHM

SERVER

Step1. Start the program

Step2. Create an unnamed socket for the server using the parameters AF_INET as domain and the SOCK_DGRAM as type.

Step3. Name the socket using bind( ) system call with the parameters server sock and the server address(sin_addr and sin_sport).

Step4. The server gets the message from the client.

Step5. Prints the message.

Step6. Stop the program

CLIENT

Step1. Start the program

Step2. Create an unnamed socket for client using socket

Step3. Call with parameters AF_INET as domain an SOCK_DGRAM as type.

Step4. Name the socket using bind( ) system call.

Step5. The Sendto( ) system call is used to deliver the Message to the server.

Step6. Stop the program

PROGRAM

SERVER

```
import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;
```

```
import java.net.SocketException;
public class udpBaseServer
{
	public static void main(String[] args) throws IOException
	{
		System.out.println("\nThis program is done by: MANISH SINGH SURYAVANSHI
\t 201061101118\n");
		DatagramSocket ds = new DatagramSocket(1234);
		byte[] receive = new byte[65535];
		DatagramPacket DpReceive = null;
		while (true)
		{
			DpReceive = new DatagramPacket(receive, receive.length);
			ds.receive(DpReceive);
			System.out.println("Client:-" + data(receive));
			if (data(receive).toString().equals("bye"))
			{
				System.out.println("Client sent bye.....EXITING");
				break;
			}
			receive = new byte[65535];
		}
	}
	public static StringBuilder data(byte[] a)
	{
		if (a == null)
			return null;
```

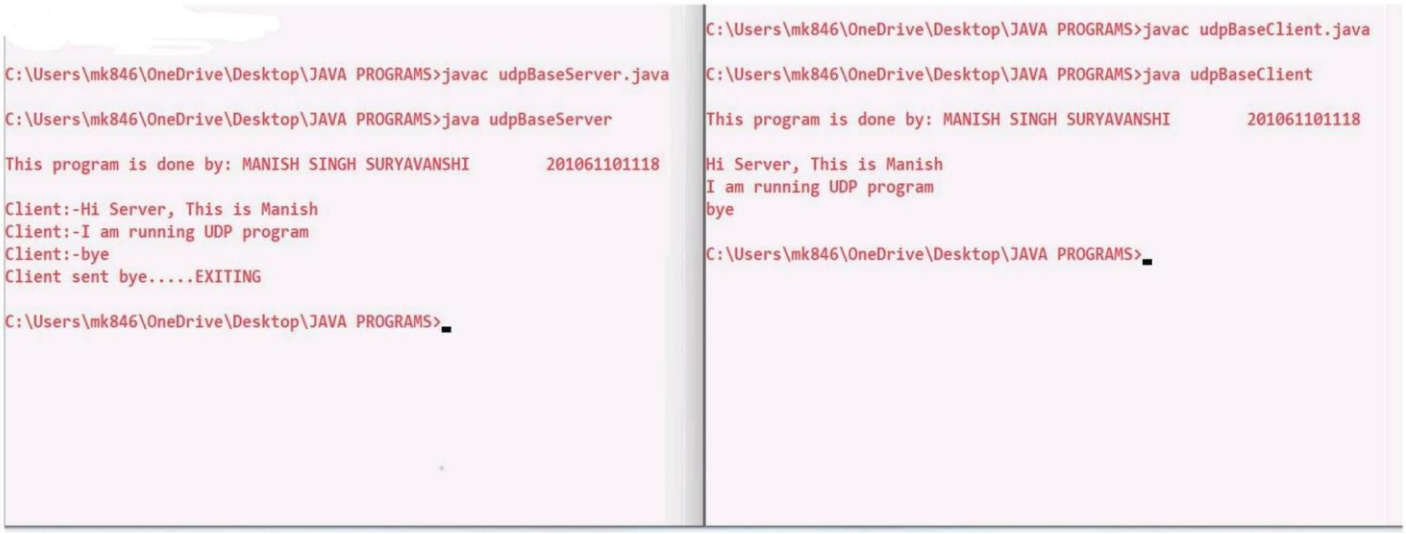
```
                StringBuilder ret = new StringBuilder();
                int i = 0;
                while (a[i] != 0)
                {
                        ret.append((char) a[i]);
                        i++;
                }
                return ret;
        }
}
```

CLIENT

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class udpBaseClient
{
        public static void main(String args[]) throws IOException
        {
                System.out.println("\nThis program is done by: MANISH SINGH SURYAVANSHI
\t 201061101118\n");
                Scanner sc = new Scanner(System.in);
                DatagramSocket ds = new DatagramSocket();
                InetAddress ip = InetAddress.getLocalHost();
                byte buf[] = null;
                while (true)
```

```
                {
                        String inp = sc.nextLine();
                        buf = inp.getBytes();
                        DatagramPacket DpSend =new DatagramPacket(buf, buf.length, ip, 1234);
                        ds.send(DpSend);
                        if (inp.equals("bye"))
                                break;
                }
        }
}
```

OUTPUT



RESULT

Thus, the given program for the connection between the client and server has been established by using UDP and executed successfully.

Ex. No: 4 **Date:**

CREATE A SOCKET (TCP) BETWEEN TWO COMPUTERS AND ENABLE FILE TRANSFER BETWEEN THEM

AIM

To write a java program for transferring a file between two computers using TCP.

ALGORITHM

SERVER

Step1. Start the program

Step2. Create an unnamed socket for the server using parameters AF_INET as domain and SOCK_STREAM as type.

Step3. Get the server port number.

Step4. Register the host address to the system by using bind() system call in server side.

Step5. Create a connection queue and wait for clients using listen() system call with the number of clients requests as parameter.

Step6. Create a Child process using fork( ) system call.

Step7. If the process identification number is equal to zero accept the connection using accept( ) system call when the client request for connection.

Step8. If pid is not equal to zero then exit the process.

Step9. Stop the program

CLIENT

Step1. Start the program

Step2. Create an unnamed socket for the client using parameters AF_INET as domain and SOCK_STREAM as type.

Step3. Get the client port number.

Step4. Now connect the socket to server using connect( ) system call.

Step5. Enter the file name.

Step6. The file is transferred from client to server using send ( ) function.

Step7. Print the contents of the file in a new file.

Step8. Stop the program

PROGRAM

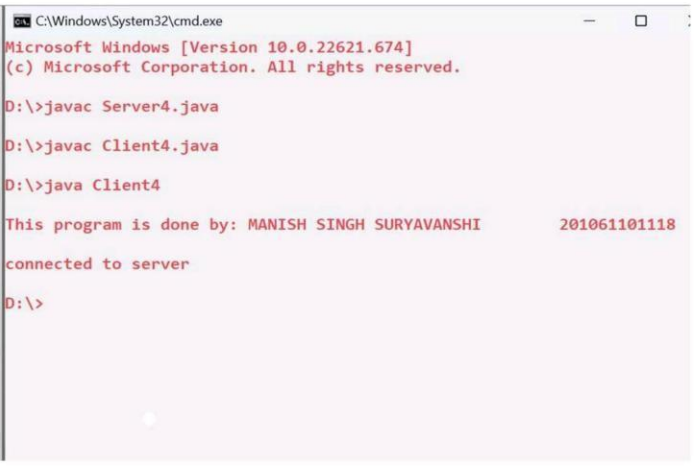
SERVER

```
import java.io.*;
import java.net.*;
class Server4
{
	public static void main(String args[]) throws IOException
	{
		System.out.println("\nThis program is done by: MANISH SINGH SURYAVANSHI
\t 201061101118\n");
		ServerSocket ss=new ServerSocket(7777);
		Socket s=ss.accept();
		System.out.println("connected...");
		FileInputStream fin=new FileInputStream("D:\\send.txt");
		DataOutputStream dout=new DataOutputStream(s.getOutputStream());
		int r;
		while((r=fin.read())!=-1)
		{
			dout.write(r);
		}
		System.out.println("\nFile transfer completed");
		s.close();
		ss.close();
	}
}
```

CLIENT

```
import java.io.*;
import java.net.*;
class Client4
{
	public static void main(String args[]) throws IOException
	{
		System.out.println("\nThis program is done by: MANISH SINGH SURYAVANSHI
\t 201061101118\n");
		Socket s=new Socket("127.0.0.1",7777);
		if(s.isConnected())
		{
			System.out.println("connected to server");
		}
		FileOutputStream fout= new FileOutputStream("D:\\received.txt");
		DataInputStream din=new DataInputStream(s.getInputStream());
		int r;
		while((r=din.read())!=-1)
		{
			fout.write((char)r);
		}
		s.close();
	}
}
```

OUTPUT



RESULT

Thus, the java program for transferring file from one machine to another machine using TCP is executed and the output is verified successfully.

EX.NO: 05 **DATE :**

Design a TCP concurrent server to echo given set of sentences using poll functions

AIM

To design and implement a TCP concurrent server that echoes a given set of sentences using poll functions in Java.

ALGORITHM

Initialize Server:

- Create a server socket and bind it to a specific port.
- Initialize a `Poller` to manage multiple socket channels.

Accept Connections:

- Use the `Poller` to monitor the server socket for incoming connections.
- Accept new connections and add them to the `Poller`.

Handle Multiple Clients:

- For each connected client, use the `Poller` to check for incoming data.
- Read data from the clients and echo the received sentences back to them.

Echo Sentences:

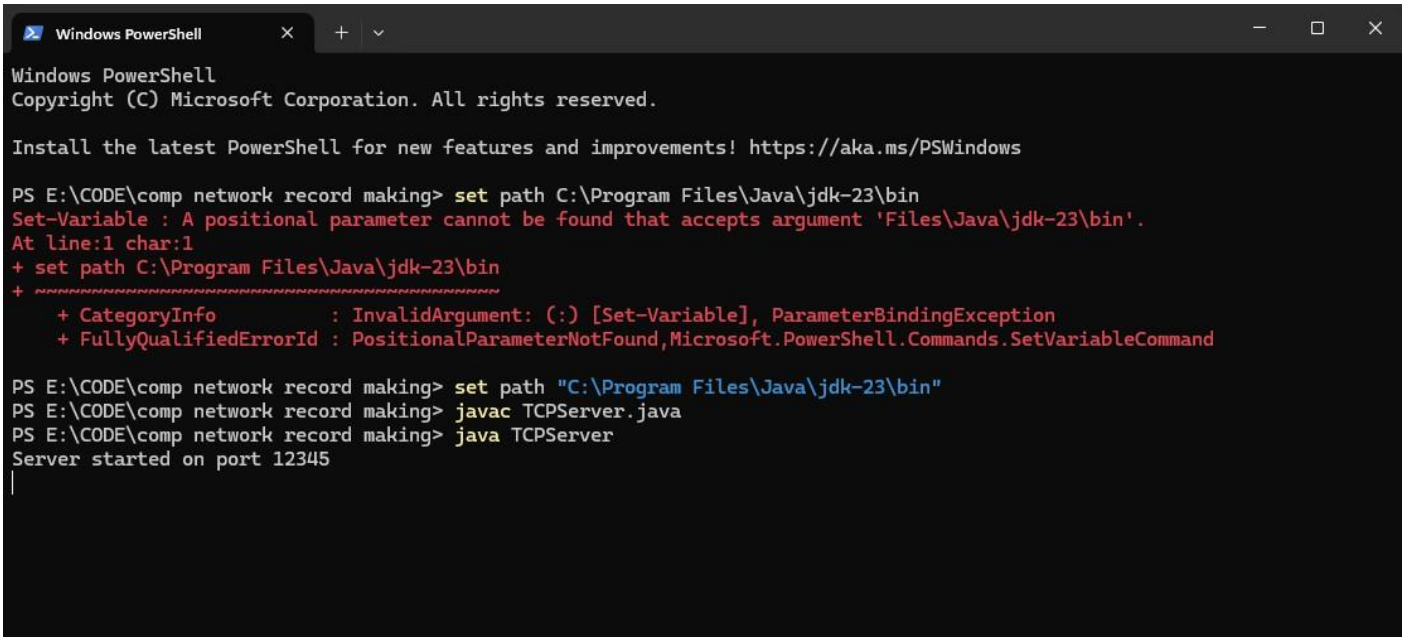
- Continuously read and write data to and from clients using non-blocking I/O operations.
- Close the connection once all sentences have been echoed.

PROGRAM

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.channels.SelectionKey;
import java.nio.channels.Selector;
import java.nio.channels.ServerSocketChannel;
import java.nio.channels.SocketChannel;
import java.net.InetSocketAddress;
import java.util.Iterator;
import java.util.Set;

public class TCPServer {
   private static final int PORT = 12345;

   public static void main(String[] args) {
      try (ServerSocketChannel serverChannel = ServerSocketChannel.open();
           Selector selector = Selector.open()) {

         serverChannel.bind(new InetSocketAddress(PORT));
         serverChannel.configureBlocking(false);
```

```
        serverChannel.register(selector, SelectionKey.OP_ACCEPT);

        System.out.println("Server started on port " + PORT);

        while (true) {
            selector.select();

            Set<SelectionKey> selectedKeys = selector.selectedKeys();
            Iterator<SelectionKey> iterator = selectedKeys.iterator();

            while (iterator.hasNext()) {
                SelectionKey key = iterator.next();

                if (key.isAcceptable()) {
                    accept(selector, serverChannel);
                } else if (key.isReadable()) {
                    readAndEcho(key);
                }

                iterator.remove();
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
  }

  private static void accept(Selector selector, ServerSocketChannel serverChannel) throws IOException {
    SocketChannel clientChannel = serverChannel.accept();
    clientChannel.configureBlocking(false);
    clientChannel.register(selector, SelectionKey.OP_READ);

    System.out.println("Accepted connection from " + clientChannel.getRemoteAddress());
  }

  private static void readAndEcho(SelectionKey key) throws IOException {
    SocketChannel clientChannel = (SocketChannel) key.channel();
    ByteBuffer buffer = ByteBuffer.allocate(1024);

    int bytesRead = clientChannel.read(buffer);
    if (bytesRead == -1) {
        clientChannel.close();
        System.out.println("Connection closed by client");
        return;
    }

    buffer.flip();
    clientChannel.write(buffer);
    buffer.clear();
  }
}
```

OUTPUT



RESULT

The TCP concurrent server using poll functions was successfully implemented in Java. The server was able to handle multiple client connections and echo the received sentences back to the clients effectively.

EX.NO: 06 **DATE :**

Implement Concurrent Time Server application using UDP to execute the program at remote server. Client sends a time request to the server sends its system time back to the client. Client displays the result

AIM

To implement a concurrent time server application using UDP, where the client sends a time request to the server, and the server sends its system time back to the client. The client then displays the result.

ALGORITHM

Initialize Server:

- Create a Datagram socket and bind it to a specific port.
- Continuously listen for incoming time requests from clients.

Handle Client Requests:

- Receive time request from the client.
- Get the current system time.
- Send the system time back to the client as a response.

Client Interaction:

- Create a Datagram socket on the client side.
- Send a time request to the server.
- Receive the system time from the server.
- Display the received time.

PROGRAM

TimeServer.java

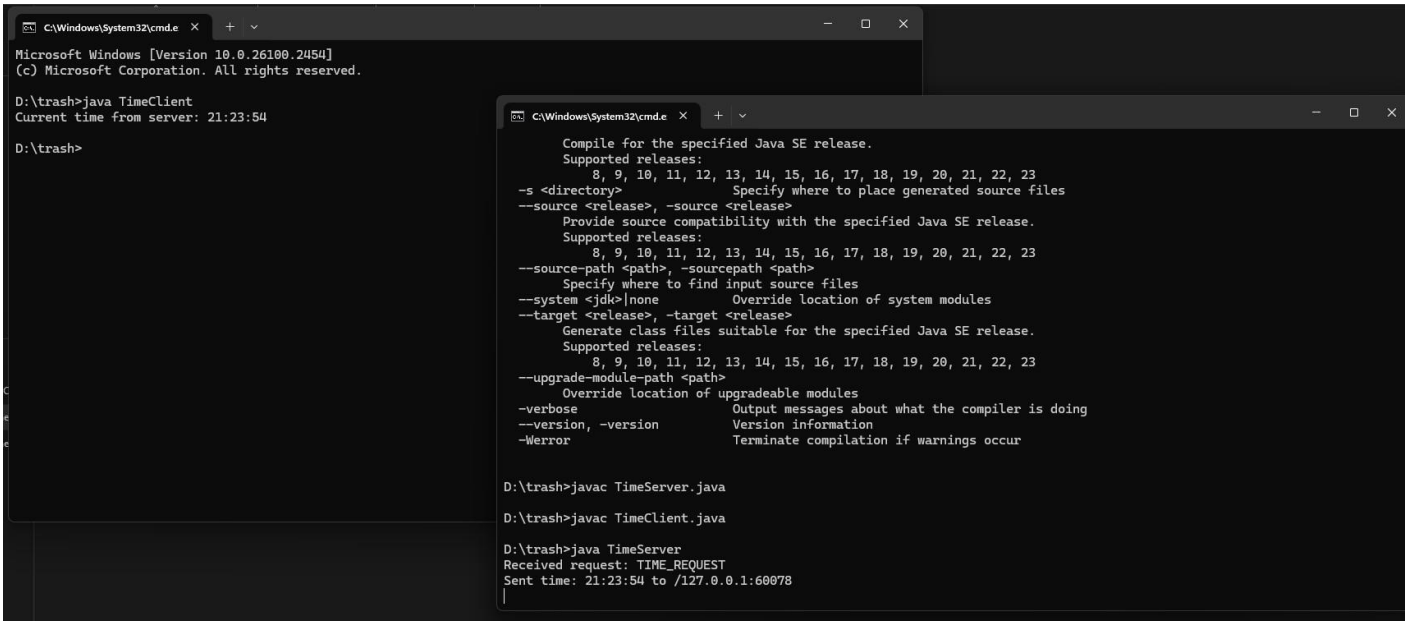
```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.text.SimpleDateFormat;
import java.util.Date;

public class TimeServer {
   public static void main(String[] args) {
      try (DatagramSocket serverSocket = new DatagramSocket(9876)) {
         byte[] receiveData = new byte[1024];
         byte[] sendData;

         while (true) {
            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
            serverSocket.receive(receivePacket);

            String request = new String(receivePacket.getData(), 0, receivePacket.getLength());
            System.out.println("Received request: " + request);
```

```
            SimpleDateFormat formatter = new SimpleDateFormat("HH:mm:ss");
            String currentTime = formatter.format(new Date());
            sendData = currentTime.getBytes();

            InetAddress clientAddress = receivePacket.getAddress();
            int clientPort = receivePacket.getPort();
            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, clientAddress,
clientPort);
            serverSocket.send(sendPacket);

            System.out.println("Sent time: " + currentTime + " to " + clientAddress + ":" + clientPort);
         }
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

TimeClient.java

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

public class TimeClient {
   public static void main(String[] args) {
      try (DatagramSocket clientSocket = new DatagramSocket()) {
         InetAddress serverAddress = InetAddress.getByName("localhost");
         byte[] sendData = "TIME_REQUEST".getBytes();
         byte[] receiveData = new byte[1024];

         DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, serverAddress,
9876);
         clientSocket.send(sendPacket);

         DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
         clientSocket.receive(receivePacket);

         String serverTime = new String(receivePacket.getData(), 0, receivePacket.getLength());
         System.out.println("Current time from server: " + serverTime);
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

OUTPUT



RESULT

Implement Concurrent Time Server application using UDP to execute the program at remote server. Client sends a time request to the server sends its system time back to the client. Client displays the result

Ex. No: 07 **Date:**

IMPLEMENTATION OF RPC IN SERVER CLIENT MODEL

AIM

To implement Remote Procedure Call (RPC) in Server-Client Model using Java.

ALGORITHM

SERVER

Step1. Start the program

Step2. Create an unnamed socket for the server.

Step3. Name the socket using bind( ) system call with the parameters server_sockfd and the server address(sin_addr and sin_sport).

Step4. The server gets the method name from the client.

Step5. Prints the output of the method.

Step6. Stop the program

CLIENT

Step1. Start the program

Step2. Create an unnamed socket for client using socket

Step3. Input the method name to be called from the server.

Step4. Pass the data to the server.

Step5. Stop the program

PROGRAM

SERVER

```
import java.io.*;

import java.net.*;

class Server5

{

        public static void main(String[] args) throws Exception

        {

                System.out.println("\nThis program is done by: MANISH SINGH SURYAVANSHI
\t 201061101118\n");
```

```
ServerSocket sersock = new ServerSocket(3000);
System.out.println("Server ready");
Socket sock = sersock.accept();
BufferedReader keyRead = new BufferedReader(new
InputStreamReader(System.in));
OutputStream ostream = sock.getOutputStream();
PrintWriter pwrite = new PrintWriter(ostream, true);
InputStream istream = sock.getInputStream();
BufferedReader receiveRead = new BufferedReader(new
InputStreamReader(istream));
String receiveMessage, sendMessage, fun;
int a, b, c;
while (true)
{
    fun = receiveRead.readLine();
    if (fun != null)
        System.out.println("Operation : " + fun);
    a = Integer.parseInt(receiveRead.readLine());
    System.out.println("Parameter 1 : " + a);
    b = Integer.parseInt(receiveRead.readLine());
    if (fun.compareTo("add") == 0)
    {
        c = a + b;
        System.out.println("Addition = " + c);
        pwrite.println("Addition = " + c);
    }
    if (fun.compareTo("sub") == 0)
```

```
                {
                        c = a - b;
                        System.out.println("Substraction = " + c);
                        pwrite.println("Substraction = " + c);
                }
                if (fun.compareTo("mul") == 0)
                {
                        c = a * b;
                        System.out.println("Multiplication = " + c);
                        pwrite.println("Multiplication = " + c);
                }
                if (fun.compareTo("div") == 0)
                {
                        c = a / b;
                        System.out.println("Division = " + c);
                        pwrite.println("Division = " + c);
                }
                System.out.flush();
            }
        }
    }
```

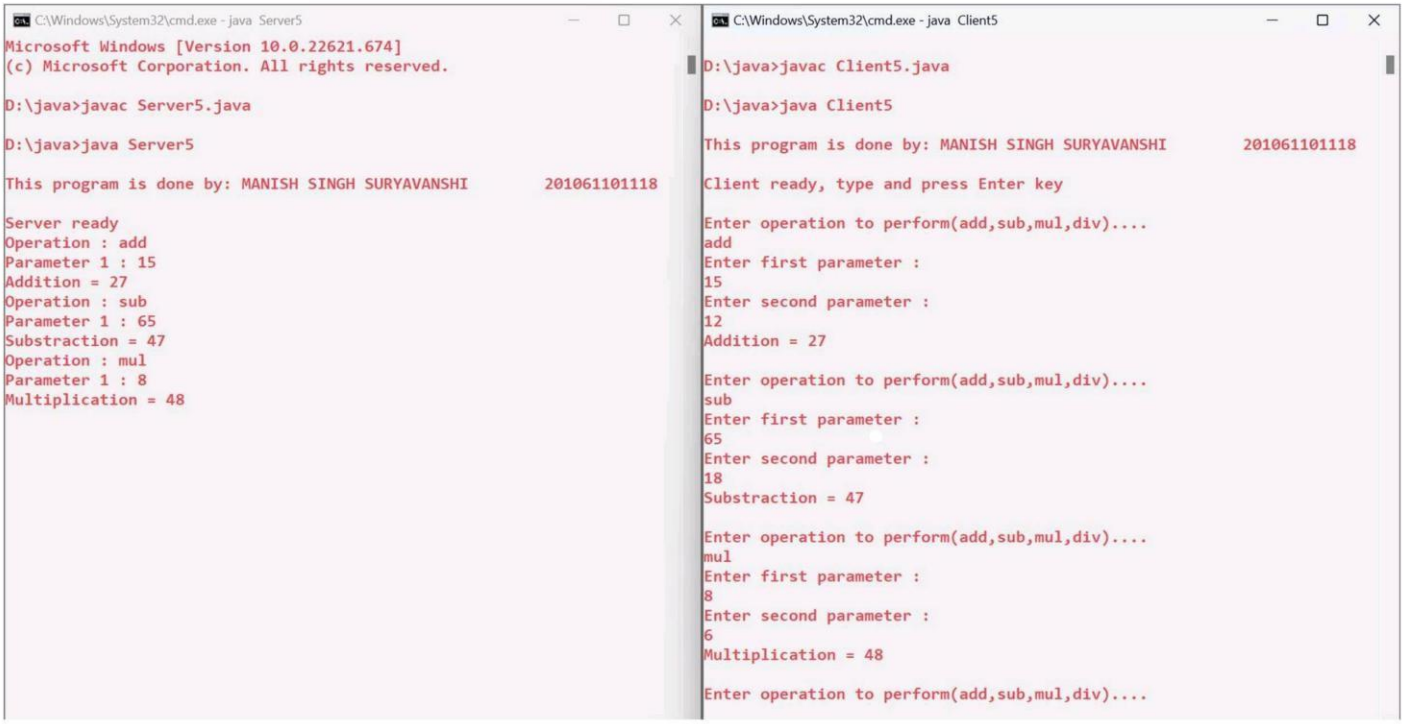
CLIENT

```
    import java.io.*;
    import java.net.*;
    class Client5
    {
```

```
        public static void main(String[] args) throws Exception
        {
                System.out.println("\nThis program is done by: MANISH SINGH SURYAVANSHI
\t 201061101118\n");
                Socket sock = new Socket("127.0.0.1", 3000);
                BufferedReader keyRead = new BufferedReader(new
InputStreamReader(System.in));
                OutputStream ostream = sock.getOutputStream();
                PrintWriter pwrite = new PrintWriter(ostream, true);
                InputStream istream = sock.getInputStream();
                BufferedReader receiveRead = new BufferedReader(new
InputStreamReader(istream));
                System.out.println("Client ready, type and press Enter key");
                String receiveMessage, sendMessage, temp;
                while (true)
                {
                        System.out.println("\nEnter operation to perform(add,sub,mul,div)....");
                        temp = keyRead.readLine();
                        sendMessage = temp.toLowerCase();
                        pwrite.println(sendMessage);
                        System.out.println("Enter first parameter :");
                        sendMessage = keyRead.readLine();
                        pwrite.println(sendMessage);
                        System.out.println("Enter second parameter : ");
                        sendMessage = keyRead.readLine();
                        pwrite.println(sendMessage);
                        System.out.flush();
```

```
                if ((receiveMessage = receiveRead.readLine()) != null)

                System.out.println(receiveMessage);

                        }

                }

        }
```

OUTPUT



RESULT

Thus, the Java program for implementing Remote Procedure Call is executed and the output is verified successfully.

Ex. No: 08 **Date:**

IMPLEMENTATION OF ARP / RARP

AIM

To write a java program for simulating ARP and RARP protocols using TCP.

ALGORITHM

SERVER

Step1. Start the program

Step2. Declare the variable.

Step3. Create a text file for input and write IP addresses

Step4. Get the IP address to be converted into MAC address.

Step5. Returns the MAC address to display.

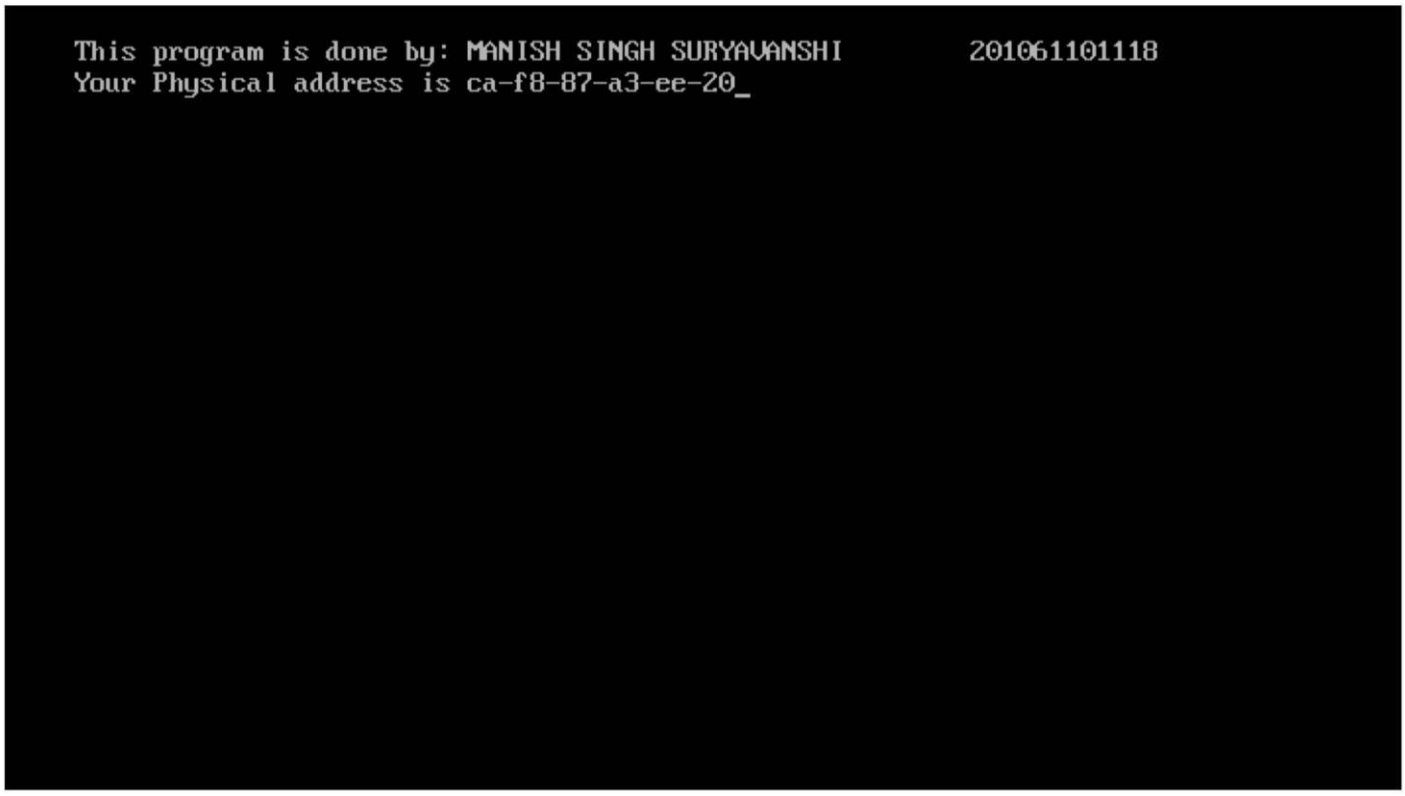
Step6. Stop the program

PROGRAM

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<conio.h>
int main()
{
        FILE *f1, *fopen();
        int ch;
        char str1[80], *token;
        const char  s[2] = " ";
        clrscr();
        printf("\nThis program is done by: MANISH SINGH SURYAVANSHI \t 201061101118\n");
        f1 = fopen("arp_rarp.txt","r");
        if ( f1 == NULL )     /* check does file exist*/
```

```
{
        printf("Cannot open file for reading \n" );
        exit(1);
}
while(fgets(str1,80,f1)!=NULL)
{
        token = strtok(str1,s);
        while (token != NULL)
        {
                if (strcmp(token,"192.168.3.2") ==0)
                {
                        token = strtok(NULL, s);
                        printf("Your Physical address is %s",token);
                }
                else
                        token = strtok(NULL, s);
        }
}
fclose(f1);
getch();
return 0;
}
```

OUTPUT



RESULT

Thus the Java program for implementing ARP/RARP is executed and the output is verified successfully.

Ex. No: 709 **Date:**

HTTP SOCKET PROGRAM TO DOWNLOAD A WEB PAGE

AIM

To write a HTTP socket program to download a web page.

ALGORITHM

Step1. Start the program

Step2. Create a SocketHTTPClient.java file and add the main method

Step3. Input the hostname and provide the default port number as 80

Step4. Create a Socket by passing hostname and port number as parameters.

Step5. Use the PrintWriter class to retrieve the HTML content

Step6. Display the content in the console.

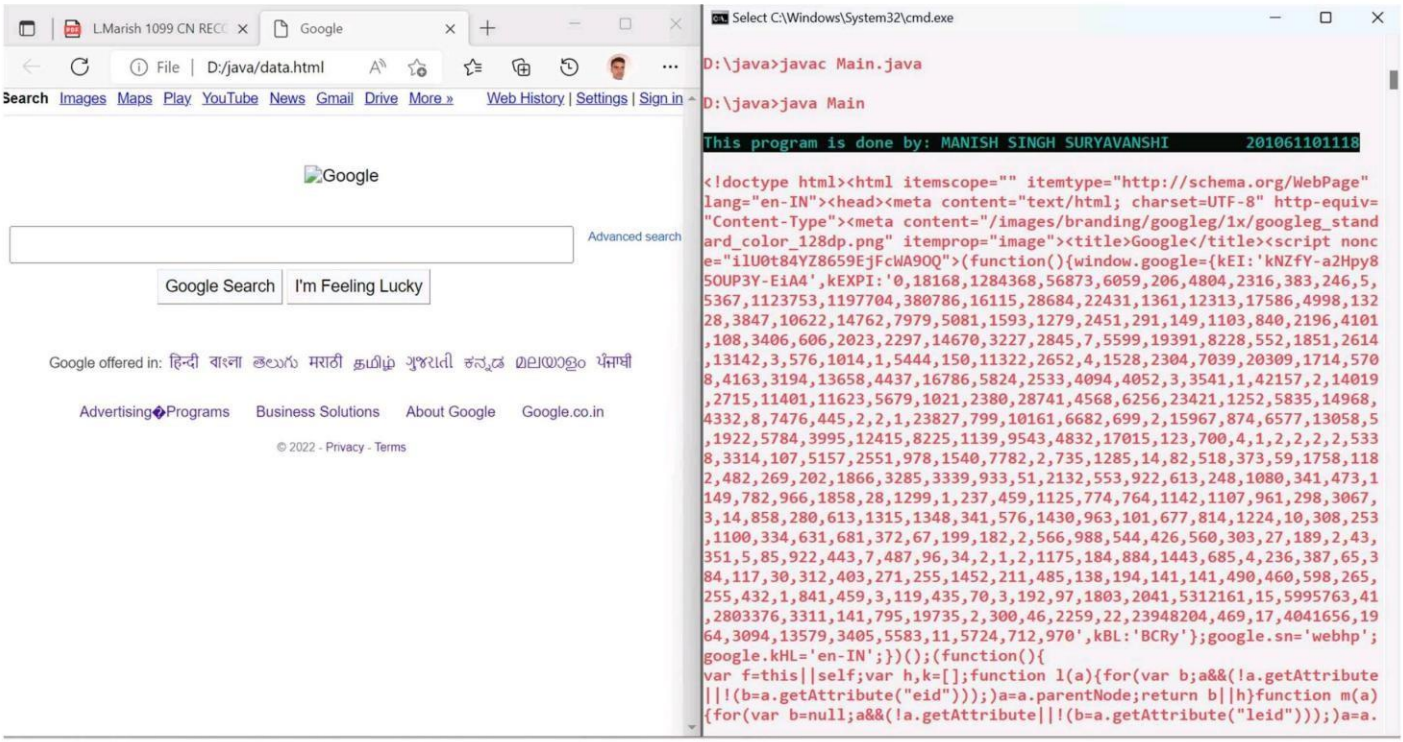
Step7. Stop the program

PROGRAM

```
import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.InputStreamReader;

import java.net.URL;

public class Main

{

        public static void main(String[] args) throws Exception

        {

                System.out.println("\nThis program is done by: MANISH SINGH SURYAVANSHI
\t 201061101118\n");

                URL url = new URL("http://www.google.com");

                BufferedReader reader=new BufferedReader(new
InputStreamReader(url.openStream()));

                BufferedWriter writer = new BufferedWriter(new FileWriter("data.html"));
```

```
                String line;
                while ((line = reader.readLine()) != null)
                {
                        System.out.println(line);
                        writer.write(line);
                        writer.newLine();
                }
                reader.close();
                writer.close();
        }
}
```

OUTPUT



RESULT

Thus, the Java program for downloading a web page using HTTP Socket is executed and the output is verified successfully.

Ex. No: 10 **Date:**

FILE TRANSFER IN CLIENT SERVER ARCHITECTURE USING RS232C

AIM

To write a java program for file transfer client-server architecture using tcp-ip protocols.

ALGORITHM

Step1. Start the program

Step2. Import java files.

Step3. To create a socket in client and server.

Step4. The client established a connection through the given port to a server.

Step5. The client accepts the connection from a server.

Step6. The client communicates with the server to transfer the files.

Step7. Exit() command is used to stop the communication for file transfer.

Step8. Stop the program

PROGRAM

```
import java.io.*;

import javax.comm.*;

public class Rs232c

{

        public static void main( String arg[] )

        {

                System.out.println("\nThis program is done by: MANISH SINGH SURYAVANSHI
\t 201061101118\n");

                try

                {

                        CommPortIdentifier ports = CommPortIdentifier.getPortIdentifier( "COM1" );

                        SerialPort port = ( SerialPort )ports.open( "RS232C", 1000 );

                        port.setSerialPortParams( 9600, SerialPort.DATABITS_8,
SerialPort.STOPBITS_1,SerialPort.PARITY_NONE );
```

```
                port.setFlowControlMode( SerialPort.FLOWCONTROL_NONE );
                OutputStream out = port.getOutputStream();
                String msg = "Transferred";
                out.write( msg.getBytes() );
                out.flush();
                out.close();
                port.close();
            }
            catch( Exception e )
            {
                System.out.println( "Error:" + e.getMessage() );
            }
        }
    }
```

OUTPUT

```
$ java Rs232c.java
Transfered
```

RESULT

Thus, the java program using RS232C has been verified and executed successfully.

Ex. No: 11 **Date:**

TO IMPLEMENT RMI (REMOTE METHOD INVOCATIO)

AIM

To implement the RMI using the given protocol.

ALGORITHM

Step1. Start the program

Step2. It initiates a connection with remove Virtual Machine (JVM).

Step3. It writes and transmits(marshals) the parameters to the remote Virtual Machine (JVM).

Step4. It waits(unmarshal) the return value or exception.

Step5. It finally, returns the value to the caller.

Step6. Stop the program

PROGRAM

SERVER

```
import java.rmi.*;
import java.rmi.server.*;
class RMIServer extends UnicastRemoteObject implements MyInterface
{
        public RMIServer()throws RemoteException
        {
                System.out.println("Remote Server is running Now.!!");
        }
        public static void main(String arg[])
        {
                System.out.println("\nThis program is done by: MANISH SINGH SURYAVANSHI
\t 201061101118\n");
                try
                {
```

```
                RMIServer p=new RMIServer();
                Naming.rebind("rmiInterface",p);
            }
            catch(Exception e)
            {
                System.out.println("Exception occurred : "+e.getMessage());
            }
        }
        public String countInput(String input) throws RemoteException
        {
            System.out.println("Received your input "+ input+"  at server!!");
            String reply;
            reply="You have typed "+ input.length() +"  letters!!";
            return reply;
        }
}
```

CLIENT

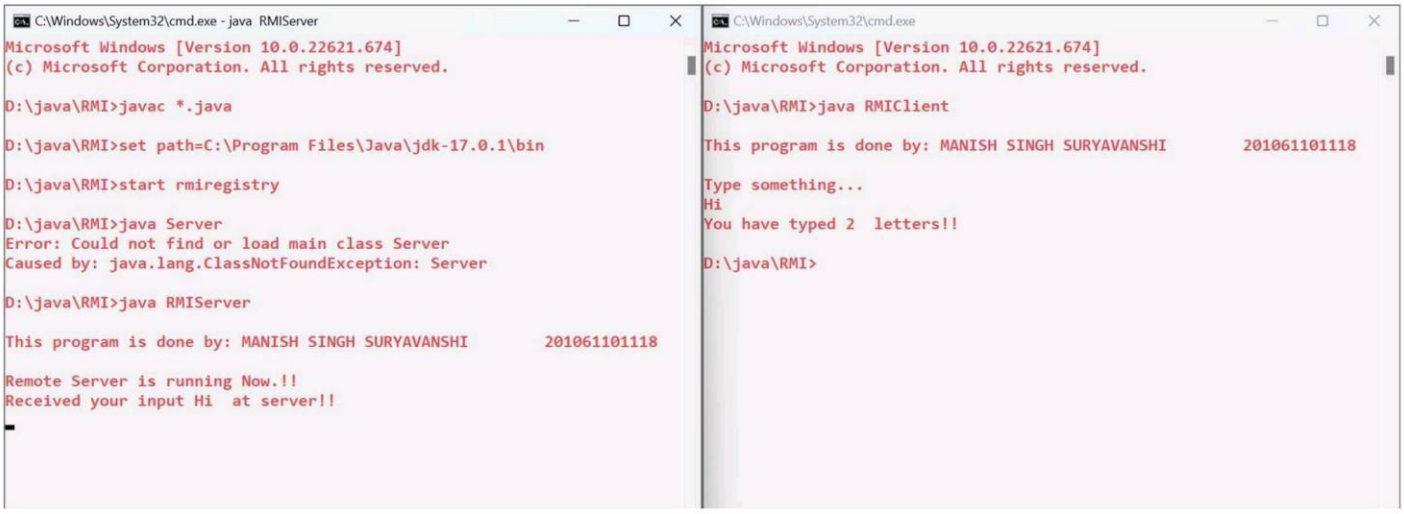
```
import java.rmi.*;
import java.io.*;
public class RMIClient
{
    public static void  main(String args[])
    {
        System.out.println("\nThis program is done by: MANISH SINGH SURYAVANSHI
\t 201061101118\n");
        try
```

```
                {
                        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                        MyInterface p=( MyInterface)Naming.lookup("rmiInterface");
                        System.out.println("Type something...");
                        String input=br.readLine();
                        System.out.println(p.countInput(input));
                }
                catch(Exception e)
                {
                        System.out.println("Exception occurred : "+e.getMessage());
                }
        }
}
```

MyInterface.java

```
import java.rmi.*;
public interface MyInterface extends Remote
{
        public String countInput(String input)throws RemoteException;
}
```

OUTPUT



RESULT

Thus, the Java program for implementing Remote Method Invocation is executed and the output is verified successfully.

Ex. No: 12 **Date:**

DEMONSTRATION OF NETWORKS SIMULATORS

AIM

To write a program for the demonstration of networks simulators.

ALGORITHM

Step1. Start the program

Step2. Import necessary packages.

Step3. Create the frame and define necessary parameters for frame.

Step4. Using socket get the local host.

Step5. Open the connection using IO Buffer stream.

Step6. Close the connection.

Step7. Stop the program

PROGRAM

```
import java.net.*;
import java.io.*;
import java.util.*;
public class GroupChat
{
        private static final String TERMINATE = "Exit";
        static String name;
        static volatile boolean finished = false;
        public static void main(String[] args)
        {
                System.out.println("\nThis program is done by: MANISH SINGH SURYAVANSHI
\t 201061101118\n");
                if (args.length != 2)
                        System.out.println("Two arguments required: <multicast-host><port-
number>");
```

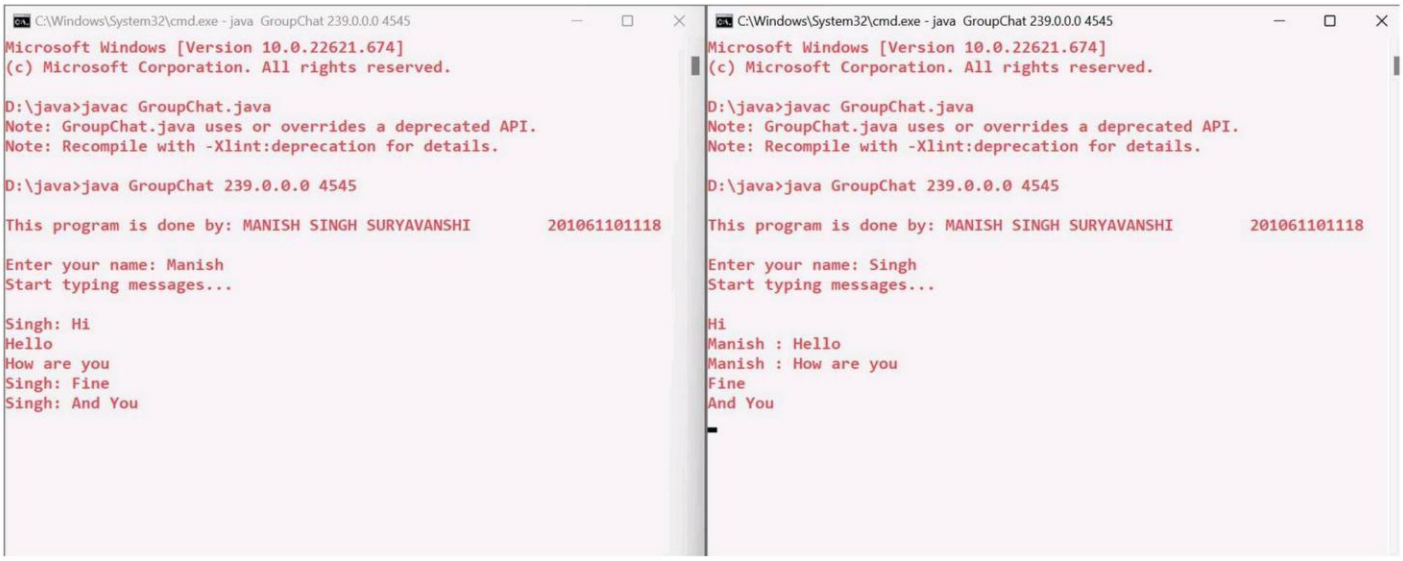
```
else
{
        try
        {
                InetAddress group = InetAddress.getByName(args[0]);
                int port = Integer.parseInt(args[1]);
                Scanner sc = new Scanner(System.in);
                System.out.print("Enter your name: ");
                name = sc.nextLine();
                MulticastSocket socket = new MulticastSocket(port);
                // Since we are deploying
                socket.setTimeToLive(0);
                //this on localhost only (For a subnet set it as 1)
                socket.joinGroup(group);
                Thread t = new Thread(new
                ReadThread(socket,group,port));
                // Spawn a thread for reading messages
                t.start();
                // sent to the current group
                System.out.println("Start typing messages...\n");
                while(true)
                {
                        String message;
                        message = sc.nextLine();
                        if(message.equalsIgnoreCase(GroupChat.TERMINATE))
                        {
```

```
                                        finished = true;
                                        socket.leaveGroup(group);
                                        socket.close();
                                        break;
                                }
                                message = name + ": " + message;
                                byte[] buffer = message.getBytes();
                                DatagramPacket datagram = new
                                DatagramPacket(buffer,buffer.length,group,port);
                                socket.send(datagram);
                        }
                }
                catch(SocketException se)
                {
                        System.out.println("Error creating socket");
                        se.printStackTrace();
                }
                catch(IOException ie)
                {
                        System.out.println("Error reading/writing from/to socket");
                        ie.printStackTrace();
                }
        }
    }
}
class ReadThread implements Runnable
```

```
{
        private MulticastSocket socket;
        private InetAddress group;
        private int port;
        private static final int MAX_LEN = 1000;
        ReadThread(MulticastSocket socket,InetAddress group,int port)
        {
                this.socket = socket;
                this.group = group;
                this.port = port;
        }
        @Override
        public void run()
        {
                while(!GroupChat.finished)
                {
                        byte[] buffer = new byte[ReadThread.MAX_LEN];
                        DatagramPacket datagram = new
                        DatagramPacket(buffer,buffer.length,group,port);
                        String message;
                        try
                        {
                                socket.receive(datagram);
                                message = new
                                String(buffer,0,datagram.getLength(),"UTF-8");
                                if(!message.startsWith(GroupChat.name))
```

```
                    System.out.println(message);
                }
                catch(IOException e)
                {
                    System.out.println("Socket closed!");
                }
            }
        }
    }
```

OUTPUT



RESULT

Thus, the java program for implementing the demonstration of network simulators has been verified and executed successfully.

EX.NO: 13 **DATE :**

DEMONSTRATION OF NETWORK SIMULATORS

AIM

To demonstrate the use of network simulators to visualize and analyze the behavior of computer networks.

ALGORITHM

Tools

For this demonstration, we'll use two popular network simulators:

1. **Cisco Packet Tracer**: A visual simulation tool that allows you to create and configure network topologies.
2. **NS-3 (Network Simulator 3)**: A discrete-event network simulator used for research and educational purposes.

Setup and Demonstration

1. Cisco Packet Tracer

Setup:

1. Download and install Cisco Packet Tracer from the official Cisco Networking Academy website.
2. Launch the application.

Demonstration:

1. **Create a Network Topology**:
   - Drag and drop networking devices (e.g., routers, switches, PCs) onto the workspace.
   - Connect the devices using cables.
2. **Configure Devices**:
   - Double-click on each device to open its configuration window.
   - Assign IP addresses and configure routing protocols as needed.
3. **Simulate Network Traffic**:
   - Use the simulation mode to create and send packets across the network.
   - Observe how packets are routed through the network and how devices communicate.

2. NS-3 (Network Simulator 3)

Setup:

1. Download and install NS-3 from the official NS-3 website or repositories.
2. Set up the environment by following the installation instructions provided on the website.

PROGRAM

Create a Simple Script:

- Create a new script file (e.g., `simple_network.cc`).
- Write a script to create a simple network topology, configure devices, and simulate traffic.

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"

using namespace ns3;

int main(int argc, char *argv[]) {
   CommandLine cmd;
   cmd.Parse(argc, argv);

   NodeContainer nodes;
   nodes.Create(2);

   PointToPointHelper pointToPoint;
   pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
   pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

   NetDeviceContainer devices;
   devices = pointToPoint.Install(nodes);

   InternetStackHelper stack;
   stack.Install(nodes);

   Ipv4AddressHelper address;
   address.SetBase("10.1.1.0", "255.255.255.0");

   Ipv4InterfaceContainer interfaces = address.Assign(devices);

   UdpEchoServerHelper echoServer(9);
   ApplicationContainer serverApps = echoServer.Install(nodes.Get(1));
   serverApps.Start(Seconds(1.0));
   serverApps.Stop(Seconds(10.0));

   UdpEchoClientHelper echoClient(interfaces.GetAddress(1), 9);
   echoClient.SetAttribute("MaxPackets", UintegerValue(1));
   echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
   echoClient.SetAttribute("PacketSize", UintegerValue(1024));

   ApplicationContainer clientApps = echoClient.Install(nodes.Get(0));
```

```
    clientApps.Start(Seconds(2.0));
    clientApps.Stop(Seconds(10.0));

    Simulator::Run();
    Simulator::Destroy();
    return 0;
}
```

OUTPUT

- Compile the script using the NS-3 build system (e.g., `waf`).
- Run the compiled script to simulate the network and observe the output.

```
Nodes created.
Devices installed.
Internet stack installed.
IP addresses assigned.
Simulation started.
Time: 1s, Sent request from Node 0 to Node 1
Time: 2s, Received response at Node 0 from Node 1
Simulation ended.
```

RESULT

The network simulators Cisco Packet Tracer and NS-3 were successfully demonstrated. These tools provide a visual and analytical understanding of network behavior, helping students learn about network topologies, configurations, and traffic simulation.