UNIVERSITY OF VISVESVARAYA COLLEGE OF ENGINEERING

A State Autonomous University on IIT Model

K.R. Circle Bengaluru-560001

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A

Practical Laboratory Manual on

Network Programming Lab

18CS1L01

Submitted for the I Semester

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY:

I Sem, M.Tech (CSE)

Batch of 2024–2026

UNDER THE GUIDANCE OF

Dr. Dilip Kumar S M

Professor

Dept of CSE, UVCE

2024–2025

Contents

Sl No	Student's Name	Register No.
1	ARFA THAREEN K	P25UV24T029001
2	ARPITHA M K	P25UV24T029002
3	BATTA NAGASAIKIRAN	P25UV24T029003
4	KAVYA R	P25UV24T029004
5	LAXMAN BASAVAYYA GONDA	P25UV24T029005
6	MEGHANA B M	P25UV24T029006
7	MOUNESH	P25UV24T029007
8	NAYAN C A	P25UV24T029008
9	NIZAMPURAM BHAVANI	P25UV24T029009
10	PANCHAMI M C	P25UV24T029010
11	RAHUL B	P25UV24T029011
12	RAKSHITH R	P25UV24T029012
13	RAMYA T	P25UV24T029013
14	SHARANPRAKASH K PATIL	P25UV24T029014
15	SRIKANTHA L	P25UV24T029015
16	V A NAVYASHREE	P25UV24T029016

GUIDE:

Dr. Dilip Kumar S M
Professor
Department of CSE
UVCE

CHAIRPERSON:

Dr. THRIVENI J
Professor
Department of CSE
UVCE

PART - A

1 Java Program to Design a TCP Iterative Client-Server Application to Reverse the Given Input Sequence

1. Server

- The server opens a `ServerSocket` on port `5000`.
- It listens in an infinite loop for incoming client connections.
- When a client connects, the server accepts the connection and uses input/output streams (`BufferedReader` and `PrintWriter`) to receive data and send the reversed result.
- After handling a single request, the server closes the connection (hence "iterative").
- The string reversal is performed using: `StringBuilder(input).reverse().toString()`.

2. Client

- The client connects to the server on `localhost` and port `5000`.
- It takes user input from the console, sends it to the server, and then waits for a response.
- Once the reversed string is received, it is displayed to the user.
- Finally, the client closes the connection.

Use Cases

This type of architecture is useful for simple **command-response models** such as chat servers, string manipulation tools, or basic file transfer protocols. It forms the foundation for more advanced topics like **multithreaded servers**, **encryption**, and **REST-based APIs**.

Client Code

```
1  import java.io.*;
2  import java.net.*;
3  class Client1
4  {
5   public static void main(String argv[]) throws Exception
6   {
7   String str="Hello networking";
8   String revStr;
9   Socket cliCon;
10  System.out.println("Connecting the Server...!");
11  cliCon = new Socket("localhost", 5555);
12
13  DataOutputStream ToServer = new DataOutputStream(cliCon.
       getOutputStream());
14  BufferedReader FromServer = new BufferedReader(new
       InputStreamReader(cliCon.getInputStream()));
15  ToServer.writeBytes(str + '\n');
16  revStr= FromServer.readLine();
17
18  System.out.println("Original String is: " + str);
19  System.out.println("Revers String is: " + revStr);
20  cliCon.close();
21  }
22 }
```

Server Code

```
1  import java.io.*;
2  import java.net.*;
3  class Server1
4  {
5   public static void main(String argv[]) throws Exception
6   {
7  String str;
8  String revStr;
9  ServerSocket ss = new ServerSocket(5555);
10
11  System.out.println("Server Running...! ");
12  Socket serCon= ss.accept();
13  BufferedReader FromClient = new BufferedReader(new
       InputStreamReader(serCon.getInputStream()));
14  DataOutputStream ToClient = new DataOutputStream(serCon.
       getOutputStream());
15  str = FromClient.readLine();
16  revStr = new StringBuilder(str).reverse().toString()+'\n';
17  ToClient.writeBytes(revStr) ;
18  ss.close();
19  serCon.close();
20  }
21  }
```

OUTPUT

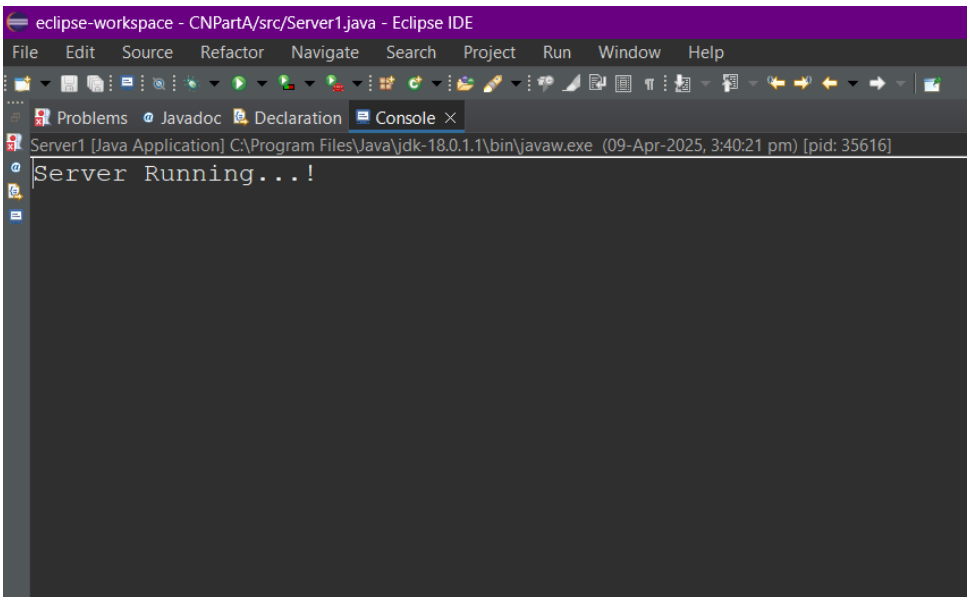


Figure 1: TCP Server

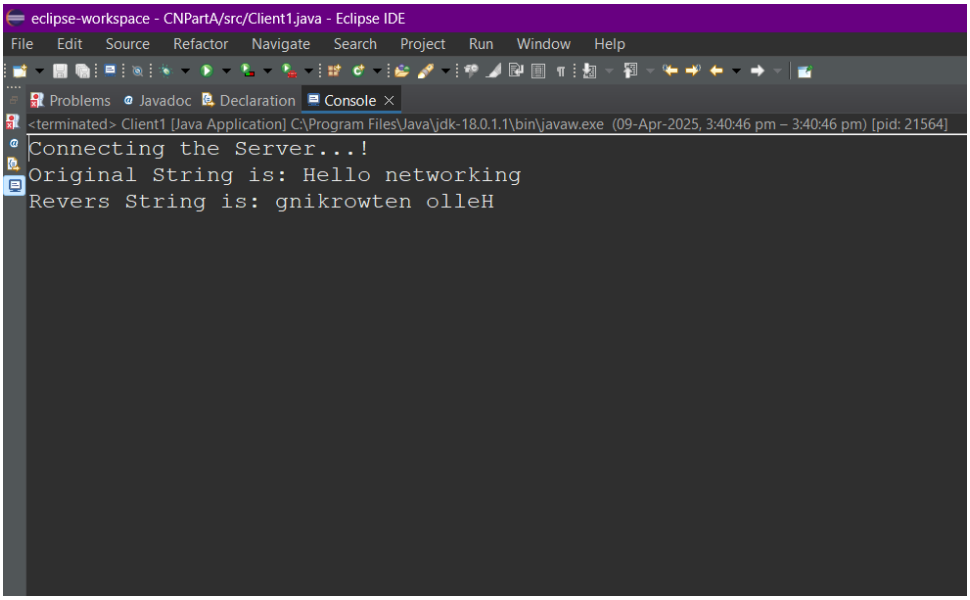


Figure 2: TCP Client

2 Write a Java program to design a TCP concurrent Client-Server application to reverse the given input sequence.

This TCP-based application demonstrates a simple client-server model where the server listens on a specific port for incoming client connections. Upon connection, the server spawns a new thread to handle that specific client, enabling multiple clients to connect and communicate concurrently. The client sends a string to the server, which the server then reverses and sends back to the client.

Key Concepts

- **TCP (Transmission Control Protocol):** Ensures reliable, ordered, and error-checked delivery of a stream of data between applications.
- **Concurrency:** Achieved using Java Threads. Each client is served by a separate thread on the server side.
- **Socket Programming:** Utilizes Java's `Socket` and `ServerSocket` classes to establish communication.

How It Works

Server Side

1. A `ServerSocket` is created and bound to a specific port (e.g., 5000).
2. The server continuously listens for incoming client connections.
3. For each new connection, a new `ClientHandler` thread is instantiated and started.
4. The thread:
    - Reads input from the client.

- Reverses the received string.
- Sends the reversed string back to the client.

Client Side

1. A `Socket` is created to connect to the server.
2. The client sends a string to the server via the socket's output stream.
3. The client reads the reversed string from the server's response and displays it.

Client Code

```
// Client.java
import java.net.*;
import java.io.*;
public class Client2
{
@SuppressWarnings({ "resource", "deprecation" })
public static void main(String[] args) throws Exception
{
while(true)
{
 Socket s=new Socket("localhost",5001);
 System.out.println("Enter String to reverse:");
 DataInputStream in=new DataInputStream(System.in);
 String str=in.readLine();
 DataOutputStream dout=new DataOutputStream(s.getOutputStream());
 dout.writeUTF(str);
 DataInputStream din=new DataInputStream(s.getInputStream());
 String rev=din.readUTF();
 System.out.println("Reversed String:\t"+rev);
}
}
}
```

Server Code

```
// Server.java
import java.net.*;
import java.io.*;
public class Server2
{
@SuppressWarnings("resource")
public static void main(String[] args) throws Exception
{
 int count=1;
 System.out.println("Server Running.....!");
 ServerSocket ss=new ServerSocket(5001);
 while(true)
 {
 new RevThread(ss.accept(),count).start();
 System.out.println(count+" Client connected.");
 count++;
 }
 }
 }
class RevThread extends Thread
{
Socket s=null;
int n;
public RevThread(Socket socket,int count)
{
 s=socket;
 n=count;
}
public void run()
{
 try
 {
 while(true)
{
 System.out.println("Receiving from client "+n);
 DataInputStream din=new DataInputStream(s.getInputStream());
 String str=din.readUTF();
 System.out.println("Processing data of Client "+n);
 String revStr=new StringBuilder(str).reverse().toString();
 System.out.println("Sending to client "+n);
 DataOutputStream dout=new DataOutputStream(s.getOutputStream());
 dout.writeUTF(revStr);
 }

```

```
45  }
46  catch(IOException e)
47  {
48
49  System.out.println(e);
50  }
51 }
52 }
```

OUTPUT

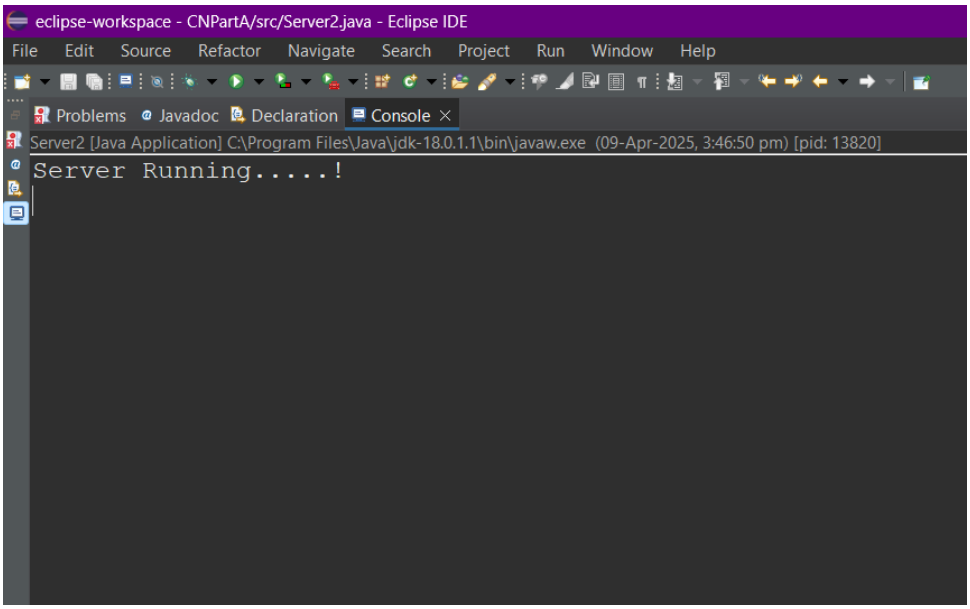


Figure 3: TCP Server

OUTPUT

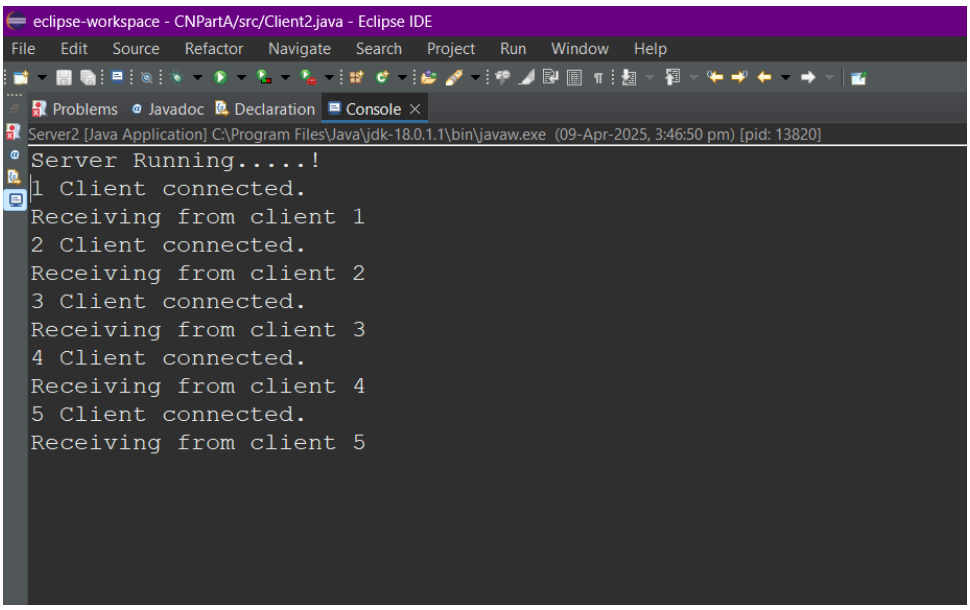


Figure 4: TCP Client

3 Write a Java program to design a TCP Client-Server application to transfer a file

This Java application demonstrates how to send a file from a client to a server using TCP sockets. The server listens on a specified port and receives files sent by the client. On the client side, a file is opened, read into a byte stream, and transmitted over the network. TCP ensures reliable and ordered delivery of data, making it ideal for file transfer operations.

Key Concepts

- **TCP (Transmission Control Protocol):** Guarantees reliable, sequential, and error-checked transmission of data between applications.
- **Socket Programming:** Employs Java's `Socket` and `ServerSocket` classes for establishing communication.
- **File I/O Streams:** Uses byte streams to read and write files across the network.

How It Works

Server Side

1. A `ServerSocket` is created and listens on a specific port (e.g., 5000).
2. When a client connects, the server accepts the incoming socket.
3. It reads the incoming byte stream from the client.
4. The received data is saved to a file (e.g., `received_file.txt`).

Client Side

1. The client connects to the server using a `Socket`.
2. A file (e.g., `sample.txt`) is opened and read into a byte array.
3. The byte stream is sent over the socket to the server.
4. Once the file is completely sent, the connection is closed.

Client Code

```
// Client.java
import java.io.*;
 import java.net.*;
 public class TCPFileClient
 {

  public static void main(String[] args) throws Exception
  {
   Socket cliCon=new Socket("localhost",4002);
   System.out.println("Connected to Server ...!");
   FileOutputStream fout=new FileOutputStream("C:/Users/moune/Desktop/
      Run.txt");
   DataInputStream din=new DataInputStream(cliCon.getInputStream());
   int r;

   while((r=din.read())!=-1)
   {
    fout.write((char)r);
   }

   fout.close();
   cliCon.close();
  }
}
```

Server Code

```
// Server.java
import java.io.*;
import java.net.*;
public class TCPFileServer
{
public static void main(String[] args) throws Exception
   {
   ServerSocket ss=new ServerSocket(4002);
   System.out.println("Server Running...!");
   Socket serCon=ss.accept();
   FileInputStream fin=new FileInputStream("C:/Users/moune/Desktop/
      transfer.txt");
   DataOutputStream dout=new DataOutputStream(serCon.getOutputStream())
      ;
   int r;
   int count=0;
   while((r=fin.read())!=-1)
   {
    dout.write(r);
    count++;
   }
   System.out.println("The capacity of the file is " +count+" bytes.");
   System.out.println("File transfer completed....!");

   fin.close();
   ss.close();
   serCon.close();
   }
}
```

OUTPUT

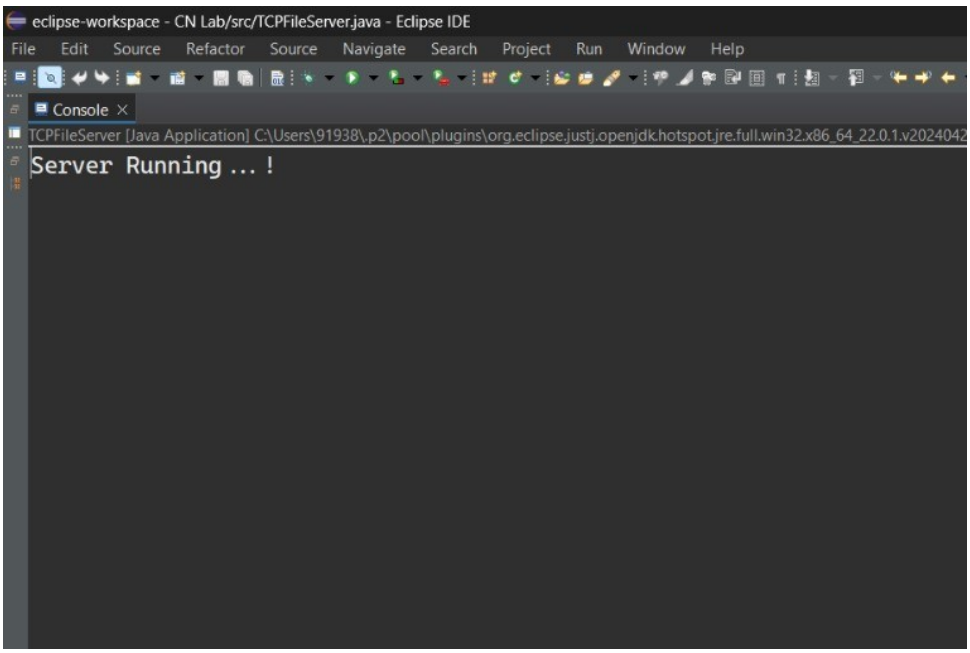


Figure 5: TCP Server

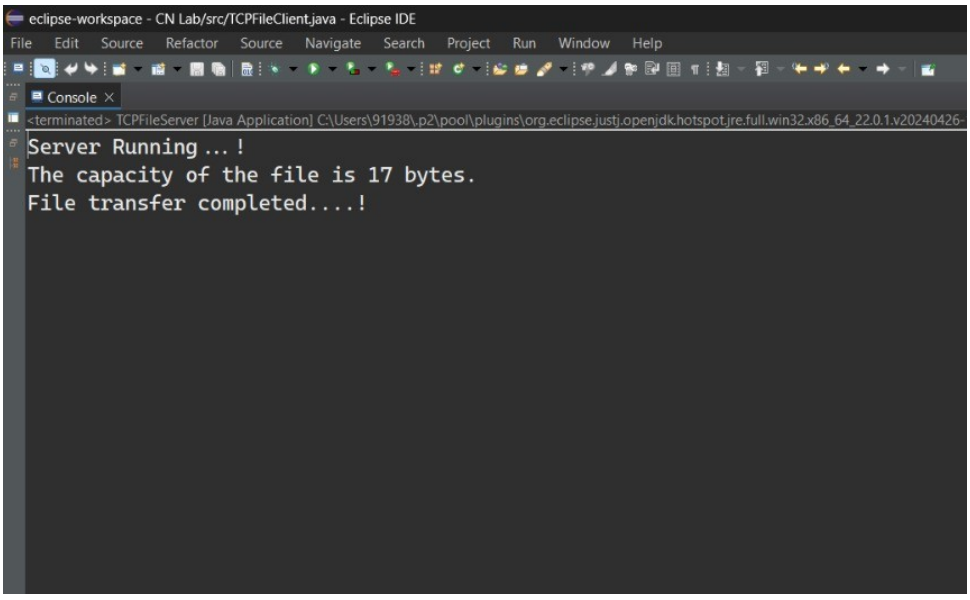


Figure 6: TCP Client

4 Write a Java program to design UDP Client-Server application to transfer a file

What is UDP?

UDP (User Datagram Protocol) is a connectionless transport protocol that transmits data as individual packets called **datagrams**, without the need to establish or maintain a connection between the sender and receiver.

It offers faster data transfer compared to TCP but does not guarantee:

- Delivery of packets
- Order of arrival
- Avoidance of duplication

Therefore, while UDP is suitable for scenarios where speed is more critical than reliability, manual mechanisms are often needed to handle lost or disordered packets.

How This Program Works

Client Side

1. Opens and reads a file (e.g., `sample.txt`) in byte chunks.
2. Each chunk is sent as a UDP packet to the server using a `DatagramSocket`.
3. After sending all data, a special packet with content "EOF" (End Of File) is transmitted to inform the server that the file transmission is complete.

Server Side

1. A `DatagramSocket` is initialized to listen on a specific port (e.g., 5000).

2. The server receives incoming UDP packets.
3. The received data is written into a new file (e.g., `received_file.txt`).
4. When the "EOF" packet is detected, the server stops receiving data.

Key Concepts Used

- **DatagramSocket:** Used for sending and receiving UDP packets.
- **DatagramPacket:** Encapsulates data to be sent or received over the network.
- **FileInputStream / FileOutputStream:** Handles reading from and writing to files as byte streams.

UDP Client Code

```
// UDPClient.java
import java.net.*;
import java.io.*;
public class UDPClient4
{
 @SuppressWarnings("resource")
 public static void main(String args[])throws Exception
 {
 byte b[]=new byte[1024];
 System.out.println("Connecting UDP Server....!");
 FileInputStream fin=new FileInputStream("C:\\Users\\moune\\Documents\\
    UDP\\UDPProgram.txt");
 DatagramSocket dsoc=new DatagramSocket();
 int i=0;
 while(fin.available()!=0)
 {
 b[i]=(byte)fin.read();
 i++;
 }
 fin.close();
 dsoc.send(new DatagramPacket(b,i,InetAddress.getLocalHost(),1000));
 }
}
```

UDP Server Code

```
// UDPServer.java
import java.net.*;
import java.io.*;
public class UDPServer4
{
 @SuppressWarnings("resource")
 public static void main(String args[])throws IOException
 {
 byte b[]=new byte[2048];
 System.out.println("UDP Server Running....!");
 DatagramSocket dsoc=new DatagramSocket(1000);
 FileOutputStream fout = new FileOutputStream("C:\\Users\\moune\\
    Documents\\UDP\\UDPProgram.txt");
 DatagramPacket dp=new DatagramPacket(b,b.length);
 dsoc.receive(dp);
 String str=new String(dp.getData());
```

```
16  byte[] strToBytes = str.getBytes();
17  fout.write(strToBytes);
18  System.out.println("File transfer completed....!");
19  fout.close();
20
21  }
22 }
```

OUTPUT

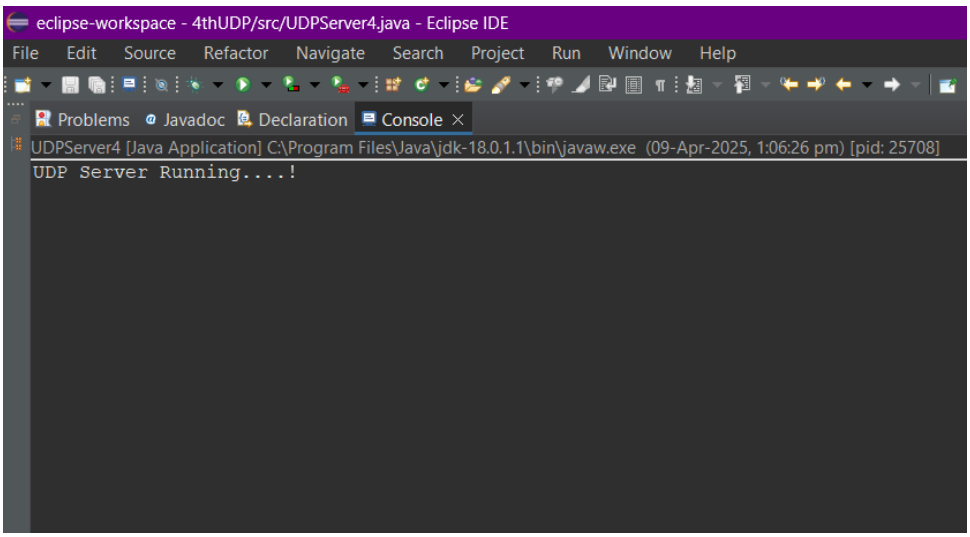


Figure 7: UDP Server

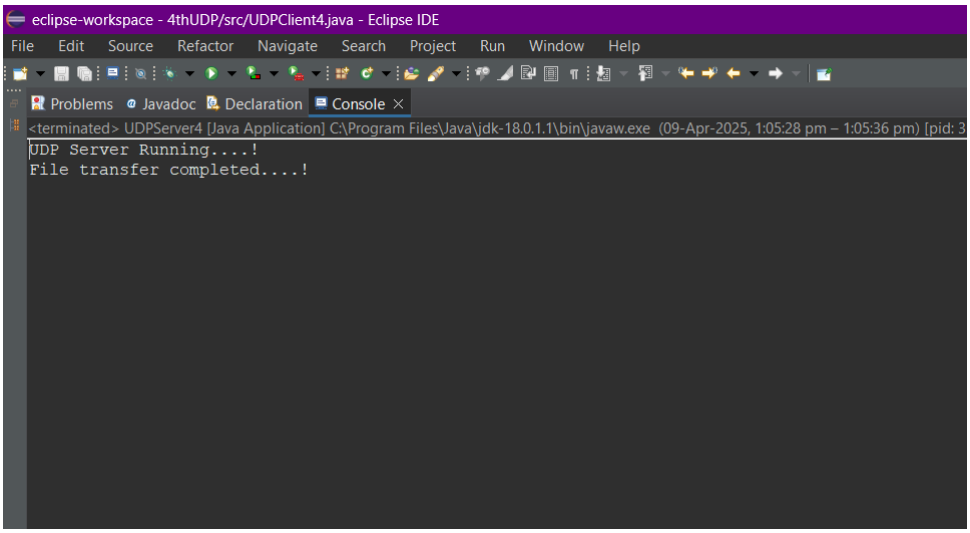


Figure 8: UDP Client

5 Write a Java program to design a ARP/RARP protocol

What is ARP?

Address Resolution Protocol (ARP) is a protocol used to map a known IP address to its corresponding MAC (Media Access Control) address on a local network.

Example

Suppose you want to send data to a device with the IP address `192.168.0.2`. Your computer will use ARP to ask:

> "Who has IP `192.168.0.2`? Tell me your MAC address."

If the device with that IP address is present on the network, it will respond with its MAC address. This MAC address is then used to deliver the packet at the data link layer.

What is RARP?

Reverse Address Resolution Protocol (RARP) is used to determine the IP address of a device when only its MAC address is known. This protocol is mainly useful for devices that do not have the capability to store IP addresses, such as diskless workstations.

Example

A diskless computer may only know its MAC address. It will broadcast a message:

> "I am MAC `AA:BB:CC:DD:EE:01`. What is my IP address?"

A RARP server on the network will respond with the appropriate IP address for that MAC address.

ARP Client Code

```java
// ARPClient.java
import java.net.*;

public class ARPServer {
    @SuppressWarnings("resource")
    public static void main(String args[]) {
        try {
            DatagramSocket sdsoc = new DatagramSocket(1309);
            while (true) {
                System.out.println("ARP Server Running...!");
                byte[] sendByte = new byte[1024];
                byte[] receiveByte = new byte[1024];
                DatagramPacket recePak = new DatagramPacket(receiveByte
                    , receiveByte.length);

                sdsoc.receive(recePak);
                String str = new String(recePak.getData());
                String s = str.trim();

                InetAddress addr = recePak.getAddress();
                int port = recePak.getPort();

                // Expanded ARP table
                String ip[] = {
                    "10.0.3.186",
                    "10.0.3.187",
                    "10.0.3.188"
                };
                String mac[] = {
                    "D4:3D:7E:12:A3:D9",
                    "00:1A:2B:3C:4D:5E",
                    "11:22:33:44:55:66"
                };

                boolean found = false;

                for (int i = 0; i < mac.length; i++) {
                    if (s.equalsIgnoreCase(mac[i])) {
                        sendByte = ip[i].getBytes();
                        DatagramPacket sendPak = new DatagramPacket(
                            sendByte, sendByte.length, addr, port);
                        sdsoc.send(sendPak);
                        found = true;
                        break;
```

```
                    }
                }

                if (!found) {
                    String msg = "MAC Address not found in ARP table.";
                    sendByte = msg.getBytes();
                    DatagramPacket sendPak = new DatagramPacket(
                        sendByte, sendByte.length, addr, port);
                    sdsoc.send(sendPak);
                }
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

ARP Server Code

```
// ARPServer.java
import java.io.*;
import java.net.*;
public class ARPClient
{
 public static void main(String args[])
 {
 try
 {
 DatagramSocket cdSoc = new DatagramSocket();
 InetAddress addr = InetAddress.getByName("localhost");
 byte[] sendByte = new byte[1204];
 byte[] receiveByte = new byte[1024];

 BufferedReader in = new BufferedReader(new InputStreamReader(System.in
    ));
 System.out.println("Enter the Physical Address: ");
 String str = in.readLine();
 sendByte = str.getBytes();

 DatagramPacket sendPak = new DatagramPacket(sendByte,sendByte.length,
    addr,1309);
 cdSoc.send(sendPak);
 DatagramPacket recePak = new DatagramPacket(receiveByte,receiveByte.
    length);
 cdSoc.receive(recePak );
```

```
24  String s = new String(recePak.getData());
25
26  System.out.println("The Logical Address is :" + s.trim());
27  cdSoc.close();
28  }
29  catch(Exception e)
30  {
31  System.out.println(e);
32  }
33  }
34 }
```

OUTPUT

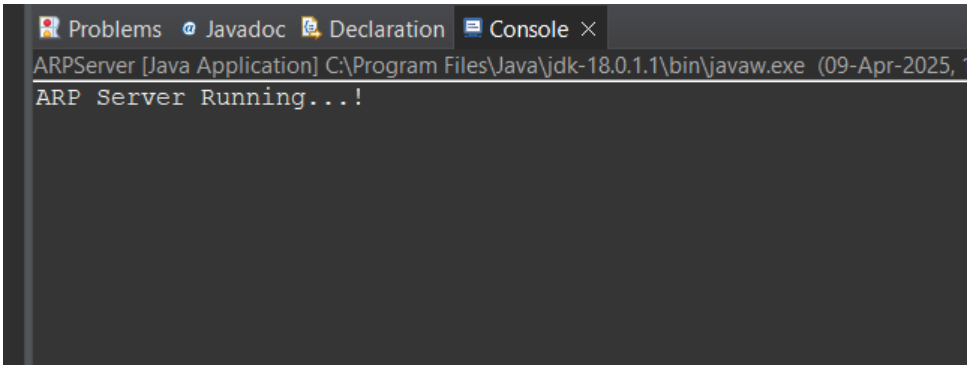


Figure 9: ARP Server

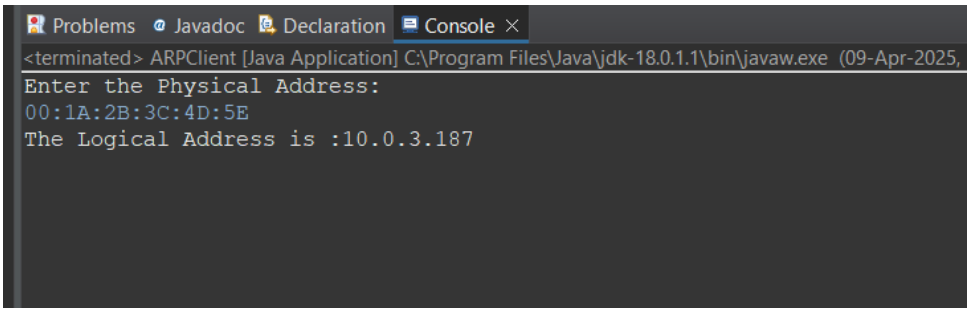


Figure 10: ARP Client

6 Write a Java program to design a DHCP protocol

Dynamic Host Configuration Protocol

The Dynamic Host Configuration Protocol (DHCP) is a network management protocol used to dynamically assign IP addresses to devices on a network. In this project, we simulate the DHCP protocol in Java using UDP sockets. While a real DHCP implementation uses raw Ethernet frames and operates at a lower level in the OSI model, this simulation provides a high-level understanding using Java's networking libraries.

Basic Idea Behind DHCP

The DHCP process typically involves the following steps:

- **DHCP Discover:** The client broadcasts a request for an IP address.
- **DHCP Offer:** The server responds with an available IP address.
- **DHCP Request:** The client accepts the offered IP address.
- **DHCP Acknowledgement:** The server confirms the assignment of the IP address.

Simulation in Java

In our Java simulation, we use UDP sockets to mimic the DHCP communication. The simulation includes:

- A **DHCP Server** that listens for incoming requests and responds with available IP addresses.
- A **DHCP Client** that sends a discover message and processes the server's response.

Concepts Used

- **DatagramSocket** and **DatagramPacket** for sending and receiving UDP packets.
- **Threads** to allow the server to continuously listen for requests while clients can operate independently.

Implementation Outline

Here is a simplified outline of how the simulation is structured:

DHCP Server

```
DatagramSocket serverSocket = new DatagramSocket(9876);
byte[] receiveData = new byte[1024];

while (true) {
    DatagramPacket receivePacket = new DatagramPacket(receiveData,
        receiveData.length);
    serverSocket.receive(receivePacket);
    // Parse message, send offer or ack
}
```

DHCP Client

```
DatagramSocket clientSocket = new DatagramSocket();
InetAddress IPAddress = InetAddress.getByName("localhost");

String discoverMessage = "DHCPDISCOVER";
byte[] sendData = discoverMessage.getBytes();

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

DHCP Client Code

```
// DHCPClient.java
import java.net.*;
public class DhcpClient
{
 public static void main(String args[])
 {
 try
 {
 DatagramSocket cdsoc = new DatagramSocket();
 byte[] sendByte = new byte[1204];
 byte[] receiveByte = new byte[1024];
 System.out.println("Sending the mac address to server : ");
 InetAddress address = InetAddress.getLocalHost();
 NetworkInterface ni = NetworkInterface.getByInetAddress(address);
 sendByte = ni.getHardwareAddress();

 for (int i = 0; i < sendByte.length; i++)
 {
 System.out.format("%02X%s", sendByte[i], (i < sendByte.length - 1) ? "
    -": "");
 }
 DatagramPacket sender = new DatagramPacket(sendByte,sendByte.length,
    address,67);
 cdsoc.send(sender);
 DatagramPacket recePak = new DatagramPacket(receiveByte,receiveByte.
    length);
 cdsoc.receive(recePak);
 String s = new String(recePak.getData());
 System.out.println("\nThe Logical Address is :" + s.trim());
 cdsoc.close();
 }
 catch(Exception e)
 {
 System.out.println(e);
 }
 }
}
```

DHCP Server Code

```
// DHCPServer.java
import java.net.*;
 import java.util.Random;
 public class DhcpServer
 {

 @SuppressWarnings("resource")
 public static void main(String args[])
 {
 try
 {
 System.out.println("DHCP Server Running...! ");
 DatagramSocket sdsoc = new DatagramSocket(67);
 while(true)
 {
 byte[] sendByte = new byte[1204];
 byte[] receiveByte = new byte[1204];
 DatagramPacket recePak = new DatagramPacket(receiveByte,receiveByte.
    length);

 sdsoc.receive(recePak);
 byte[] mac=recePak.getData();

 System.out.print("Server allocating IP Address to:");
 for (int i = 0; i <=5; i++)
 {
 System.out.format("%02X%s", mac[i], (i <5) ? "-": "");
 }

 InetAddress addr = recePak.getAddress();
 int port = recePak.getPort();
 String ip[] = {"10.0.3.186","10.0.3.187","10.0.3.188","10.0.3.189","
    10.0.3.189"};

 Random rand = new Random();
 int i = rand.nextInt(4);
 ip[i]=ip[i]+" lease time->24hrs";
 System.out.println("->"+ ip[i]);

 sendByte = ip[i].getBytes();
 DatagramPacket sendPak = new DatagramPacket(sendByte,sendByte.length,
    addr,port);
 sdsoc.send(sendPak);

```

```
42  }
43  }
44  catch(Exception e)
45  {
46  System.out.println(e);
47  }
48  }
49  }
```

OUTPUT

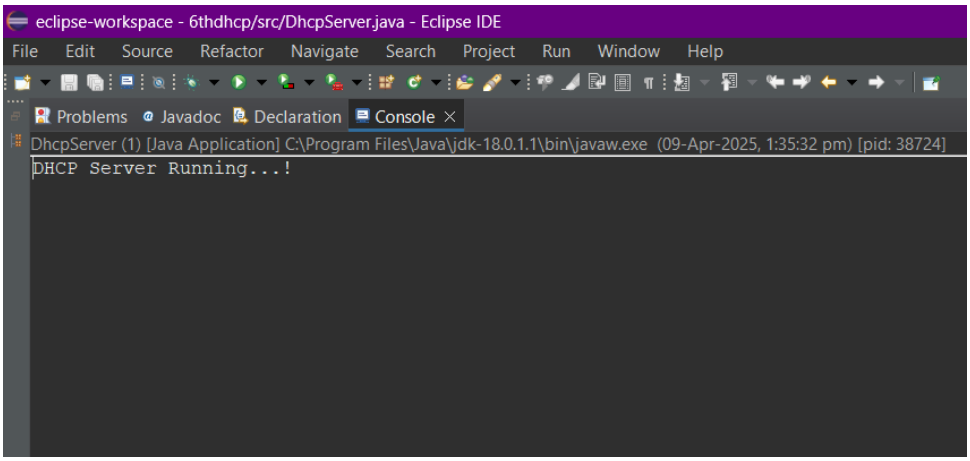


Figure 11: DHCP Server

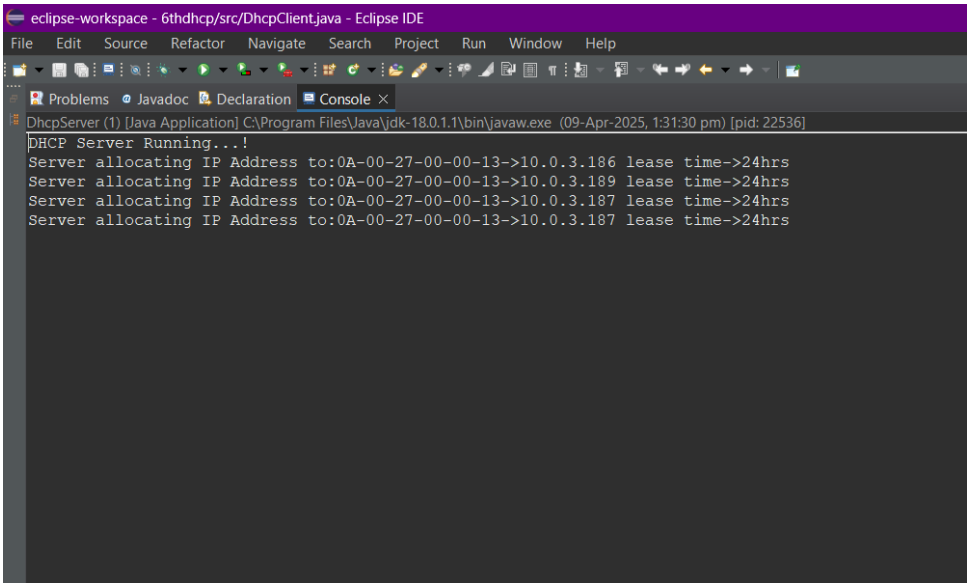


Figure 12: DHCP Client

7 Write a Java program to Distance Vector Routing protocol

Distance Vector Routing Protocol

The Distance Vector Routing Protocol is a dynamic routing algorithm used in computer networks. Each router maintains a routing table that stores the best-known distance (cost) to reach every other router and the next hop to take. It is a *distributed protocol*, meaning that routers update their tables independently based on periodic information exchanges with their immediate neighbors.

How It Works

- Each router periodically shares its routing table with directly connected neighbors.
- Upon receiving a neighbor's routing table, a router updates its own table using the **Bellman-Ford algorithm**.
- If the cost to reach a destination via a neighbor is lower than the current cost, the routing table is updated.
- Over time, all routers converge to the shortest paths to every destination in the network.

Key Concepts

Distance Vector: Each router maintains a vector (table) of distances to all possible destinations and the corresponding next hops.

Bellman-Ford Algorithm: A dynamic programming algorithm used to compute the shortest paths from one node to all others.

Cost Matrix: Represents direct link costs between routers; a very large number (e.g., 9999) indicates no direct link.

Next Hop: The neighbor to which packets are forwarded to reach a particular destination efficiently.

Java Program Highlights

The Java simulation models this protocol for a fixed number of routers.

Input

- Number of routers in the network.
- Cost matrix indicating the direct link cost (e.g., hop count or latency) between routers.

Routing Table Update

- Each router uses nested loops to simulate receiving distance vectors from all other routers.
- The Bellman-Ford algorithm is used to update the values in the distance and next hop tables.
- Specifically, the program updates `distance[i][j]` and `nextHop[i][j]` for all routers i and j.

Output

- The final routing table for each router is printed.
- Each routing table displays:
  - Destination router
  - Next hop
  - Distance (cost)

```
// DistanceVectorRouting.java
import java.util.Scanner;
public class DistanceVectorRouting
{
 private int distances[];
 private int ver;
 public static final int MAX_VALUE = 999;
 public DistanceVectorRouting(int numofver)
 {
 this.ver = numofver;
 distances = new int[numofver + 1];
 }
 public void BellmanFordEvaluation(int source, int matrix[][])
 {
 for (int node = 1; node <= ver; node++)
 {
 distances[node] = MAX_VALUE;
 }
 distances[source] = 0;
 for (int node = 1; node <= ver - 1; node++)
 {
 for (int src = 1; src <= ver; src++)
 {
 for (int dest = 1; dest <= ver; dest++)
 {
 if (matrix[src][dest] != MAX_VALUE)
 {
 if (distances[dest] > distances[src]
 + matrix[src][dest])
 distances[dest] = distances[src]
 + matrix[src][dest];
 }
 }
 }
 }
 for (int vertex = 1; vertex <= ver; vertex++)
 {
 System.out.println("distance from source " + source + " to "
 + vertex + " is " + distances[vertex]);
 }
 }
 public static void main(String... arg)
 {
 int numofver = 0;
 int source;
 Scanner scanner = new Scanner(System.in);
 System.out.println("Enter the number of vertices");
```

```
48  numofver = scanner.nextInt();
49  int matrix[][] = new int[numofver + 1][numofver + 1];
50  System.out.println("Enter the adjacency matrix");
51  for (int src = 1; src <= numofver; src++)
52  {
53  for (int dest = 1; dest <= numofver; dest++)
54  {
55  matrix[src][dest] = scanner.nextInt();
56  if (src == dest)
57  {
58  matrix[src][dest] = 0;
59  continue;
60  }
61  if (matrix[src][dest] == 0)
62  {
63  matrix[src][dest] = MAX_VALUE;
64  }
65  }
66  }
67  System.out.println("Enter the source vertex");
68  source = scanner.nextInt();
69  DistanceVectorRouting dvr = new DistanceVectorRouting(numofver);
70  dvr.BellmanFordEvaluation(source, matrix);
71  scanner.close();
72  }
73 }
```

OUTPUT

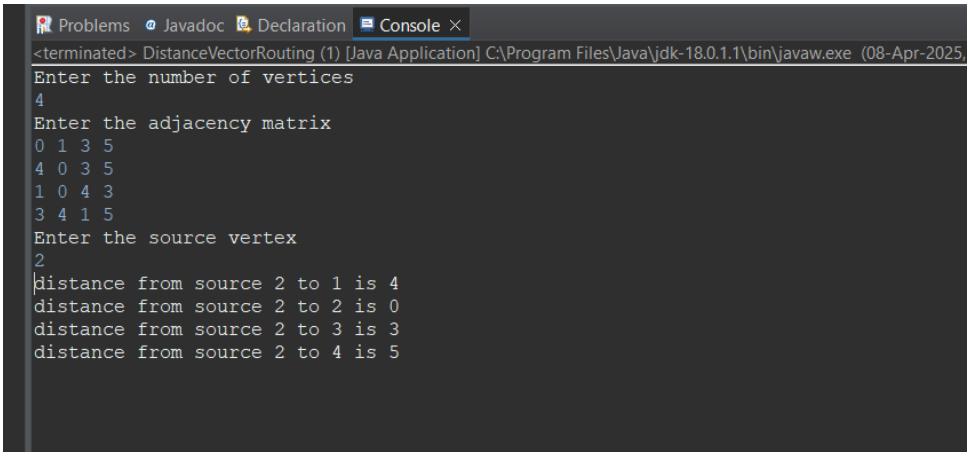


Figure 13: Distance Vector Routing

8 Write a Java program to Dijkstra's Shortest Path Routing protocol

Dijkstra's Algorithm

Dijkstra's algorithm is a graph-based routing algorithm used to find the shortest path from a source router to all other routers in a network. It is commonly implemented in link-state routing protocols, such as **OSPF (Open Shortest Path First)**.

How It Works

Initialization

- Set the distance to the **source node** as 0.
- Set the distance to all other nodes as ∞ (infinity).
- Mark all nodes as **unvisited**.

Visit Nodes

- Repeatedly select the unvisited node with the smallest known distance.
- For each unvisited neighbor of the current node:
  - Calculate the tentative distance through the current node.
  - If the new distance is smaller than the previously recorded distance, update it.
- Mark the current node as **visited**.
- Repeat the process until all nodes are visited.

Java Program Highlights

The Java implementation of Dijkstra's algorithm includes the following features:

- Accepts a **cost matrix**, where `graph[i][j]` represents the cost between routers i and j.
- Uses an array `dist[]` to track the shortest known distance from the source router to every other router.
- A helper function `minDistance()` identifies the unvisited router with the smallest tentative distance.
- Updates each neighbor's distance if a shorter path through the current router is found.

```
// DijShortestPath.java
import java.util.*;
public class Dijk
{
public int distance[] = new int[10];
public int cost[][] = new int[10][10];
public void calc(int n,int s)
{
int flag[] = new int[n+1];
int i,minpos=1,k,c,minimum;

for(i=1;i<=n;i++)
{
flag[i]=0;
this.distance[i]=this.cost[s][i];
}
c=2;
while(c<=n)
{
minimum=99;
for(k=1;k<=n;k++)
{
if(this.distance[k]<minimum && flag[k]!=1)
{
minimum=this.distance[i];
minpos=k;
}
}
 flag[minpos]=1;
 c++;
 for(k=1;k<=n;k++)
 {
 if(this.distance[minpos]+this.cost[minpos][k] < this.distance[k] &&
    flag[k]!=1 )
 this.distance[k]=this.distance[minpos]+this.cost[minpos][k];
 }
}

}
@SuppressWarnings("resource")
public static void main(String args[])
{
int nodes,source,i,j;
Scanner in = new Scanner(System.in);
System.out.println("Enter the Number of Nodes \n");
nodes = in.nextInt();
Dijk d = new Dijk();
```

```
47 System.out.println("Enter the Cost Matrix Weights: \n");
48  for(i=1;i<=nodes;i++)
49  for(j=1;j<=nodes;j++)
50
51  {
52  d.cost[i][j]=in.nextInt();
53  if(d.cost[i][j]==0)
54  d.cost[i][j]=999;
55  }
56  System.out.println("Enter the Source Vertex :\n");
57  source=in.nextInt();
58
59  d.calc(nodes,source);
60  System.out.println("The Shortest Path from Source \t"+source+"\t to
       all other vertices are : \n");
61  for(i=1;i<=nodes;i++)
62  if(i!=source)
63  System.out.println("source :"+source+"\t destination :"+i+"\t MinCost
       is :"+d.distance[i]+"\t");
64
65 }
66 }
```

OUTPUT

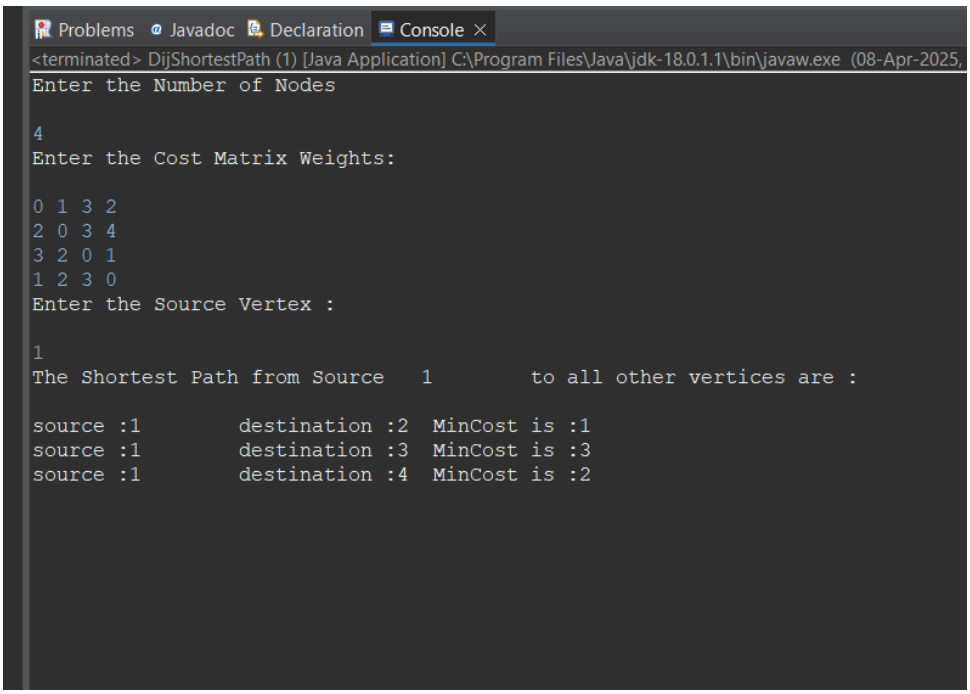


Figure 14: Dijkstra's

PART - B
Network Simulator-3

Network Simulator (NS-3)

The `ns-3` (Network Simulator 3) is an open-source discrete-event network simulator designed for research and educational purposes. It enables users to simulate how packet-based networks operate, providing detailed models for Internet protocols and other networking mechanisms.

The primary advantage of `ns-3` lies in its ability to simulate real-world networking scenarios in a controlled environment. This makes it especially valuable for:

- Conducting experiments that are difficult or costly to perform on actual hardware.
- Testing and evaluating protocols in a reproducible setting.
- Understanding the behavior of complex network systems.

While `ns-3` emphasizes Internet protocol modeling, it also supports simulations for non-Internet-based networks, making it flexible for a wide range of research areas.

Figure 14 illustrates the general execution flow of an `ns-3` simulation program. The simulation begins with configuring the network topology, including nodes, devices, channels, and protocol stacks. Applications (e.g., traffic generators) are then installed on the nodes, and simulation parameters such as timing and tracing are set. The simulator is run using a discrete-event engine, which processes all scheduled events. Finally, the results can be analyzed through trace files or visualized using tools like NetAnim.

This structured approach enables researchers to thoroughly analyze network performance, behavior under load, protocol interactions, and fault scenarios — all without the need for a physical testbed.

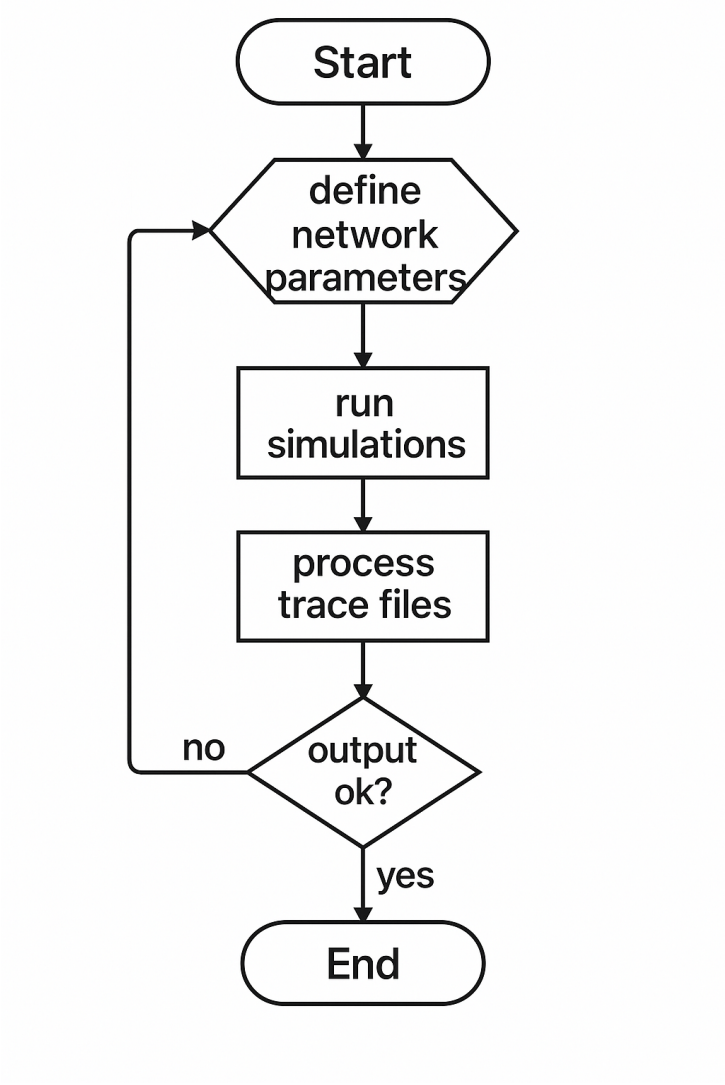


Figure 15: Flow chart of a typical `ns-3` program execution process.

Distinguishing Features of NS-3

Several network simulators exist for conducting simulation-based research and performance evaluations. However, `ns-3` stands out due to the following distinguishing features:

1. `ns-3` is designed as a collection of modular libraries, allowing integration with other external software tools and libraries. Unlike simulation platforms that provide an all-in-one graphical user interface (GUI), `ns-3` promotes modularity and extensibility.
2. External tools can be used with `ns-3` for animation, data analysis, and visualization. This includes tools like NetAnim and PyViz for simulation visualization, and Gnuplot or Python-based tools for post-simulation analysis.
3. Users interact with `ns-3` primarily through the command line, and simulations are written

in C++ and/or Python.

4. `ns-3` is primarily designed for Linux systems, though it also supports FreeBSD and Cygwin for Windows. Native Windows support via Visual Studio is under development.

Important Points about NS-3

- `ns-3` is not backward compatible with `ns-2`; it is a complete redesign built from the ground up to replace `ns-2`.
- The core simulation engine is written in C++, and Python can optionally be used as an interface for scripting and configuration.
- Many architectural limitations and bugs present in `ns-2` are addressed in `ns-3`, making it more robust and modern.
- The number of contributed or community-developed modules in `ns-3` is currently smaller compared to `ns-2`, but it is steadily growing.
- `ns-2`'s use of a dual-language architecture (C++ and OTcl) made debugging more difficult. In contrast, `ns-3`'s single-language design (C++ with optional Python) simplifies development and debugging.
- `ns-3` offers an emulation mode, enabling interaction with real systems and live networks, which enhances its applicability in hybrid simulation-emulation environments.

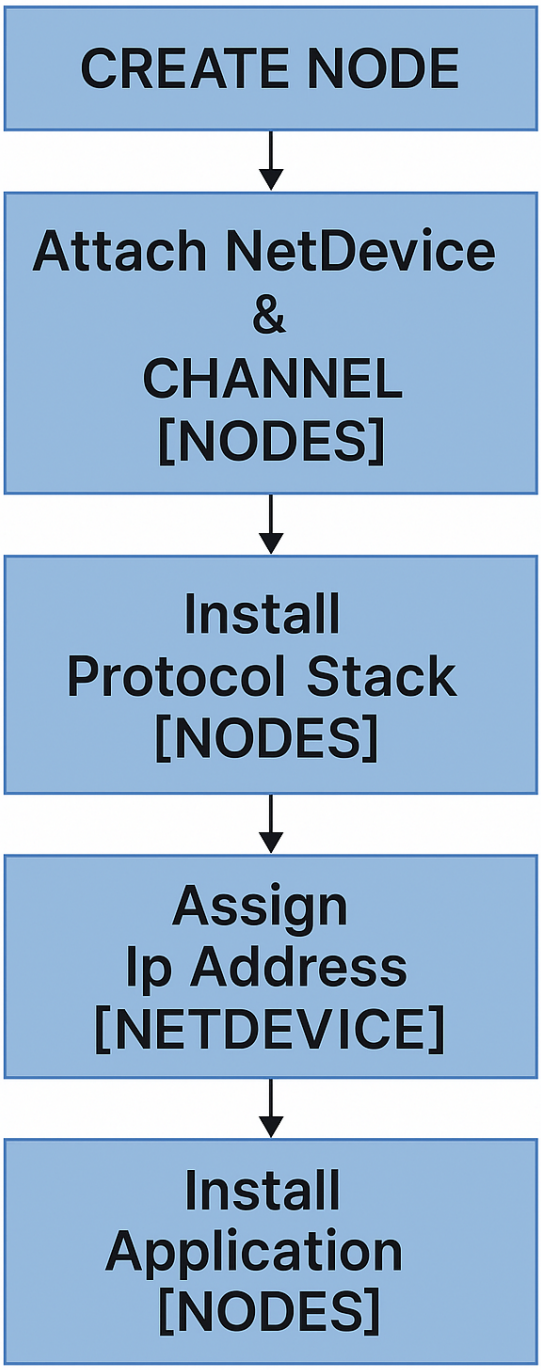


Figure 16: NS -3 Program Structure flow chart

Steps to Install NS-3 (Version 3.27)

To install NS-3 on an Ubuntu system, follow the steps below:

Step 1: Open the Terminal

Once Ubuntu is installed, open the terminal using the shortcut:

```
Ctrl + Alt + T
```

Step 2: Install Prerequisite Packages

Update the system and install all required dependencies by executing the following commands:

```
sudo apt-get update
sudo apt-get install gcc g++ python
sudo apt-get install mercurial python-setuptools git
sudo apt-get install qt5-default
sudo apt-get install python-pygraphviz python-kiwi python-pygoocanvas
libgoocanvas-
dev ipython
sudo apt-get install openmpi-bin openmpi-common openmpi-doc libopenmpi-dev
sudo apt-get install autoconf cvs bzr unrar
sudo apt-get install gdb valgrind
sudo apt-get install uncrustify
sudo apt-get install doxygen graphviz imagemagick
sudo apt-get install texlive texlive-extra-utils texlive-latex-extra \
                     texlive-font-utils texlive-lang-portuguese dvipng
sudo apt-get install python-sphinx dia
sudo apt-get install gsl-bin libgsl2 libgsl-dev
sudo apt-get install flex bison libfl-dev
sudo apt-get install tcpdump
sudo apt-get install sqlite sqlite3 libsqlite3-dev
sudo apt-get install libxml2 libxml2-dev
sudo apt-get install cmake libc6-dev libc6-dev-i386 libclang-dev
sudo pip install cxxfilt
sudo apt-get install libgtk2.0-0 libgtk2.0-dev
sudo apt-get install vtun lxc
sudo apt-get install libboost-signals-dev libboost-filesystem-dev
```

Step 3: Download NS-3 (Version 3.27)

Create a directory for NS-3 and download the release:

```
mkdir ns3
cd ns3
wget https://www.nsnam.org/release/ns-allinone-3.27.tar.bz2
tar xjf ns-allinone-3.27.tar.bz2
cd ns-allinone-3.27/
ls
```

You will find a file named `build.py` along with other necessary files.

Step 4: Build NS-3 with Examples and Tests

Run the following command to build NS-3 along with its examples and tests:

```
./build.py --enable-examples --enable-tests
```

If the build is successful, you will see the message:

```
    Build finished successfully
```

Step 5: Configure with Waf Build Tool

Navigate to the NS-3 directory and configure it using the Waf build system:

```
cd ns-3.27
./waf -d debug --enable-examples --enable-tests configure
```

Step 6: Build with Waf (Optional)

To build the simulator using Waf:

```
./waf
```

Step 7: Verify the Installation

Run the test script to ensure everything is working correctly:

```
./test.py
```

If all tests pass successfully, your NS-3 installation is complete!

NS-3 installation done successfully

Steps to Install NetAnim

To visualize NS-3 network simulations, NetAnim can be installed using the following steps:

Step 1: Open the Terminal

Open your terminal using the keyboard shortcut:

```
Ctrl + Alt + T
```

Step 2: Install Required Dependencies

Install the necessary Qt4 and XML libraries by running:

```
sudo apt-get install qt4-qmake
sudo apt-get install libqt4-dev
sudo apt-get install libxml2-dev
```

Step 3: Navigate to the NetAnim Directory

Change the directory to where NetAnim is located within your NS-3 package:

```
cd ns3/ns-allinone-3.27/netanim-3.108
```

Step 4: Build NetAnim

Clean any previous builds, configure the project, and compile using the following commands:

```
make clean
qmake NetAnim.pro
make
```

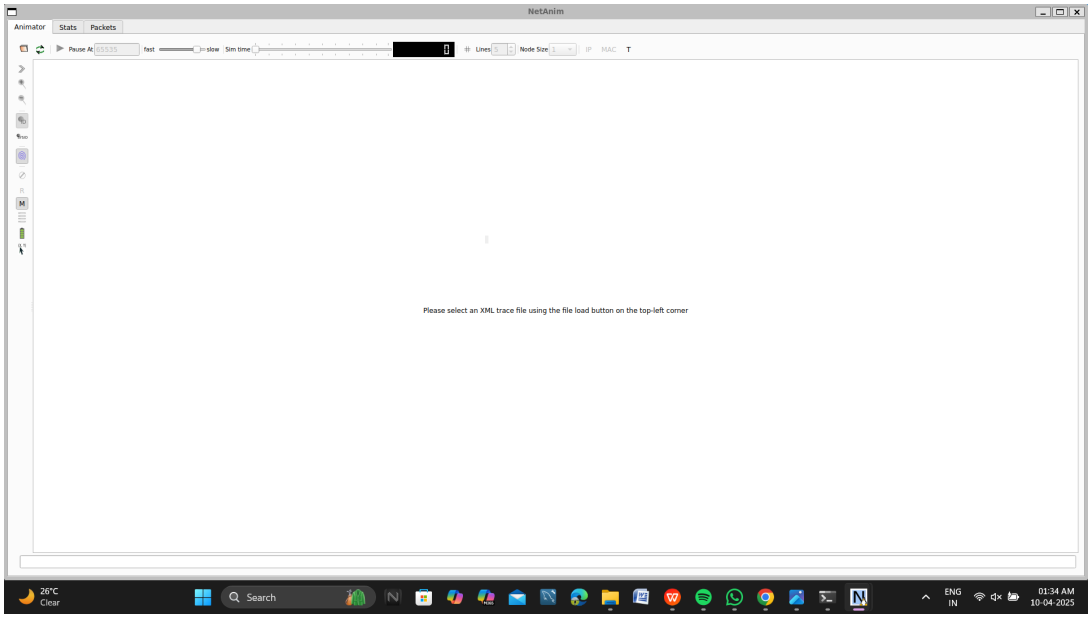
Step 5: Run NetAnim

Once the build is successful, launch NetAnim with:

```
./NetAnim
```

NetAnim should now be ready to use for visualizing NS-3 simulations.

SNAPSHOT



Steps to Install TraceMetrics

TraceMetrics is a Java-based tool used for analyzing trace files generated by NS-2 and NS-3. Follow the steps below to install and run it:

Step 1: Open the Terminal

Use the shortcut to open your terminal:

```
Ctrl + Alt + T
```

Step 2: Download TraceMetrics

Visit the official SourceForge page to download the tool:

```
http://sourceforge.net/projects/tracemetrics/
```

Step 3: Extract the Archive

After downloading, open the `.zip` file using the Archive Manager and extract the files to your desired location.

Step 4: Navigate to the TraceMetrics Directory

In the terminal, navigate to the extracted directory:

```
cd tracemetrics-1.4.0/
```

Step 5: Run TraceMetrics

Execute the following command to launch TraceMetrics:

```
java -jar "tracemetrics.jar"
```

The GUI interface of TraceMetrics should now open, allowing you to load and analyze NS simulation trace files.

9 Write a C++ program to connect two nodes on NS-3 (for practice only)

A simple star-like topology in NS-3 involving three nodes. One node acts as the central node, while the other two are connected directly to it. that connects two nodes using a point-to-point link. This is the simplest form of communication topology used in NS-3 and serves as a foundation for more complex simulations.

Network Topology

The network contains:

- Node0: One end of the communication (e.g., client)
- Node1: The other end (e.g., server)

The connection is as follows:

$$\text{Node0} \leftrightarrow \text{Node1}$$

Simulation Details

Node Creation

Two nodes are created using the NS-3 NodeContainer class.

Link Configuration

A single point-to-point link is set up between Node0 and Node1 with specified attributes like data rate (e.g., 5Mbps) and delay (e.g., 2ms).

Internet Stack and Addressing

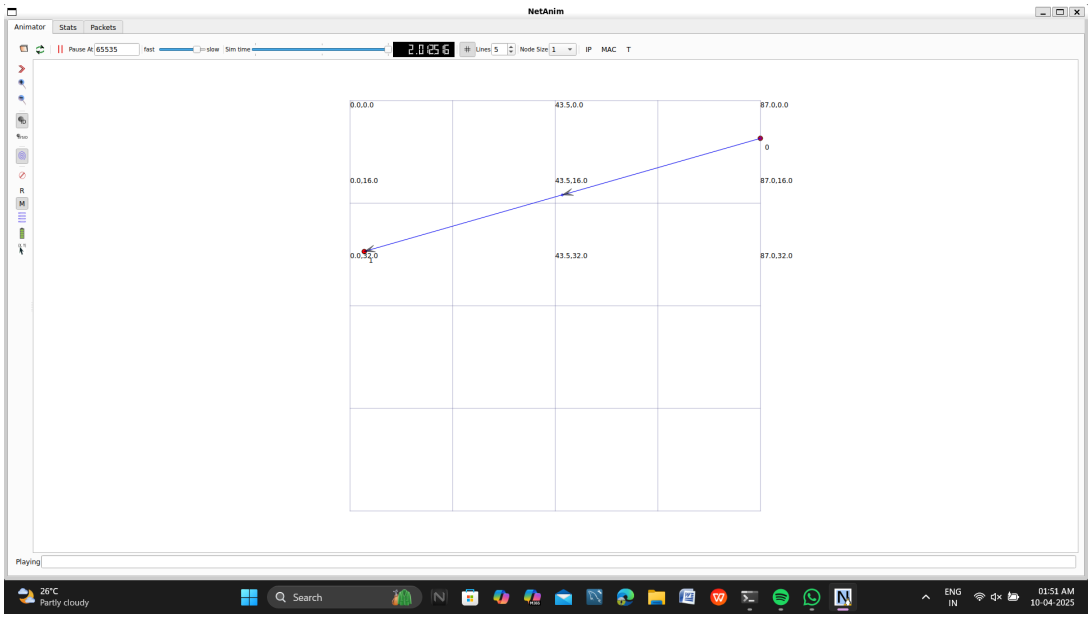
Both nodes are installed with the Internet stack. The point-to-point link is assigned a subnet (e.g., 10.1.1.0/24) and each node receives an IP address.

```
#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"
#include "ns3/netanim-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

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

    // Log settings
    Time::SetResolution (Time::NS);
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

    // Create two nodes
    NodeContainer nodes;
    nodes.Create (2);

    // Setup point-to-point channel
    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("1Mbps"))
        ;
    pointToPoint.SetChannelAttribute ("Delay", StringValue ("1ms"));

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

    // Install internet stack
    InternetStackHelper stack;
    stack.Install (nodes);

    // Assign IP addresses
    Ipv4AddressHelper address;
    address.SetBase ("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfaces = address.Assign (devices);

    // Setup UDP echo server on node 1
    UdpEchoServerHelper echoServer (9);
```

```
46      ApplicationContainer serverApps = echoServer.Install (nodes.Get (1)
            );
47      serverApps.Start (Seconds (1.0));
48      serverApps.Stop (Seconds (10.0));
49
50      // Setup UDP echo client on node 0
51      UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
52      echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
53      echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
54      echoClient.SetAttribute ("PacketSize", UintegerValue (2048));
55      ApplicationContainer clientApps = echoClient.Install (nodes.Get (0)
            );
56      clientApps.Start (Seconds (2.0));
57      clientApps.Stop (Seconds (10.0));
58
59      // Setup animation and tracing
60      AnimationInterface anim("np1.xml");
61      AsciiTraceHelper eventTraces;
62      pointToPoint.EnableAsciiAll(eventTraces.CreateFileStream("np1.tr"))
            ;
63      pointToPoint.EnablePcapAll ("np1");
64
65      Simulator::Run ();
66      Simulator::Destroy ();
67      return 0;
68  }
```

Listing 1: Simple NS-3 Script to Connect Two Nodes

OUTPUT


NetAnim
Animator
Stats
Packets
0.0.0.0
43.5.0.0
87.0.0.0
0
0.0.16.0
43.5.16.0
87.0.16.0
0.0.32.0
2
43.5.32.0
87.0.32.0
Playing
26°C
Partly cloudy
Search
ENG
IN
01:51 AM
10-04-2025

10 Write a C++ program to connect three nodes considering one as a central node on NS-3 (for practice only)

In a topology in NS-3. Node 1 acts as the central node connecting node 0 and node 2 via different point-to-point links. The UDP echo server runs on node 2, and the client runs on node 0.

Network Topology

The network consists of:

- Node0: Peripheral node (could be a client)
- Node1: Central node (acts as a hub or router)
- Node2: Peripheral node (could be a server)

The connections are:

$$\text{Node0} \leftrightarrow \text{Node1} \leftrightarrow \text{Node2}$$

Each connection uses a point-to-point link.

Simulation Details

Node Creation

Three nodes are created. Node1 is treated as the central node, with point-to-point connections to both Node0 and Node2.

Link Configuration

Two point-to-point links are created:

- Between Node0 and Node1
- Between Node1 and Node2

Each link can have attributes such as 5Mbps data rate and 2ms delay.

Internet Stack and Addressing

All three nodes are installed with the Internet stack. Each point-to-point link is assigned a unique IP subnet.

Routing

Global routing is enabled so packets can be routed from Node0 to Node2 via Node1. The following NS-3 program demonstrates how to connect two nodes using a point-to-point link, configure a UDP echo server and client, and trace the simulation with NetAnim and PCAP.

```
#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"
#include "ns3/netanim-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");

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

    Time::SetResolution (Time::NS);
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

    // Create three nodes
    NodeContainer nodes;
    nodes.Create (3);

    // Setup two point-to-point links
    PointToPointHelper p2p1;
    p2p1.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
    p2p1.SetChannelAttribute ("Delay", StringValue ("1ms"));

    PointToPointHelper p2p2;
    p2p2.SetDeviceAttribute ("DataRate", StringValue ("10Mbps"));
    p2p2.SetChannelAttribute ("Delay", StringValue ("5ms"));

    // Install Internet stack
    InternetStackHelper stack;
    stack.Install (nodes);

    // Assign IP addresses and install devices
    Ipv4AddressHelper address;
    NetDeviceContainer devices;

    // First link: node 0 <--> node 1
    address.SetBase ("10.1.1.0", "255.255.255.0");
    devices = p2p1.Install (nodes.Get (0), nodes.Get (1));
    Ipv4InterfaceContainer interfaces = address.Assign (devices);

    // Second link: node 1 <--> node 2
```

```
48      address.SetBase ("10.1.2.0", "255.255.255.0");
49      devices = p2p2.Install (nodes.Get (1), nodes.Get (2));
50      interfaces = address.Assign (devices);
51
52      // Populate routing tables
53      Ipv4GlobalRoutingHelper::PopulateRoutingTables();
54
55      // Install UDP echo server on node 2
56      UdpEchoServerHelper echoServer (9);
57      ApplicationContainer serverApps = echoServer.Install (nodes.Get (2)
            );
58      serverApps.Start (Seconds (1.0));
59      serverApps.Stop (Seconds (10.0));
60
61      // Install UDP echo client on node 0 targeting node 2
62      UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
63      echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
64      echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
65      echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
66      ApplicationContainer clientApps = echoClient.Install (nodes.Get (0)
            );
67      clientApps.Start (Seconds (2.0));
68      clientApps.Stop (Seconds (10.0));
69
70      // Animation and tracing
71      AnimationInterface anim("np2.xml");
72      AsciiTraceHelper eventTraces;
73      p2p1.EnableAsciiAll(eventTraces.CreateFileStream("np2.tr"));
74      p2p1.EnablePcapAll ("np2");
75
76      Simulator::Run ();
77      Simulator::Destroy ();
78      return 0;
79  }
```

Listing 2: Three Nodes with One Central Node in NS-3

OUTPUT

11 Write a C++ program to implement a star topology on NS-3

A star topology with one central hub and multiple spokes using NS-3. Each spoke communicates with the hub node using TCP applications. A star topology consists of a central node (hub or switch) that is directly connected to multiple peripheral nodes.

Network Topology

In a star topology:

- One central node is connected to all other nodes.
- Peripheral nodes communicate with each other through the central node.
- The failure of the central node results in network breakdown, but individual node failures do not affect the network.

Simulation Details

Node Creation

Multiple peripheral nodes (e.g., five) and one central node are created. The central node serves as a communication hub.

Link Configuration

Each peripheral node is connected to the central node using individual point-to-point links. These links can be configured with attributes like data rate and delay.

Internet Stack and Addressing

The Internet stack is installed on all nodes. Each link is assigned a unique subnet, and IP addresses are configured accordingly.

Routing

Global routing is used to populate routing tables. This ensures that any peripheral node can communicate with another through the central node.

Applications

A UDP Echo Server is installed on one peripheral node, and a UDP Echo Client is installed on another. Communication is routed via the central node.

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

// Network Star topology (default)
//
//         n2 n3 n4              .
//          \ | /                .
//           \|/                 .
//      n1--- n0---n5            .
//           /|\                 .
//          / | \                .
//         n8 n7 n6              .
//


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("Star");

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

  //
  // Set up some default values for the simulation.
  //
  Config::SetDefault ("ns3::OnOffApplication::PacketSize",
     UintegerValue (137));

  // ??? try and stick 15kb/s into the data rate
  Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue ("
     14kb/s"));
  // Default number of nodes in the star.  Overridable by command line
     argument.

  uint32_t nSpokes = 8;
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
  PointToPointStarHelper star (nSpokes, pointToPoint);

  NS_LOG_INFO ("Install internet stack on all nodes.");
```

```
45  InternetStackHelper internet;
46  star.InstallStack (internet);
47
48  NS_LOG_INFO ("Assign IP Addresses.");
49  star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0", "
       255.255.255.0"));
50
51  NS_LOG_INFO ("Create applications.");
52  //
53  // Create a packet sink on the star "hub" to receive packets.
54  //
55  uint16_t port = 50000;
56  Address hubLocalAddress (InetSocketAddress (Ipv4Address::GetAny (),
       port));
57  PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory",
       hubLocalAddress);
58  ApplicationContainer hubApp = packetSinkHelper.Install (star.GetHub
       ());
59  hubApp.Start (Seconds (1.0));
60  hubApp.Stop (Seconds (10.0));
61
62  //
63  // Create OnOff applications to send TCP to the hub, one on each
       spoke node.
64  //
65  OnOffHelper onOffHelper ("ns3::TcpSocketFactory", Address ());
66  onOffHelper.SetAttribute ("OnTime", StringValue ("ns3::
       ConstantRandomVariable[Constant=1]"));
67  onOffHelper.SetAttribute ("OffTime", StringValue ("ns3::
       ConstantRandomVariable[Constant=0]"));
68
69  ApplicationContainer spokeApps;
70
71  for (uint32_t i = 0; i < star.SpokeCount (); ++i)
72    {
73      AddressValue remoteAddress (InetSocketAddress (star.
           GetHubIpv4Address (i), port));
74      onOffHelper.SetAttribute ("Remote", remoteAddress);
75      spokeApps.Add (onOffHelper.Install (star.GetSpokeNode (i)));
76    }
77  spokeApps.Start (Seconds (1.0));
78  spokeApps.Stop (Seconds (10.0));
79
80  NS_LOG_INFO ("Enable static global routing.");
81  //
82  // Turn on global static routing so we can actually be routed across
       the star.
```

```
83   //
84   Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
85
86   NS_LOG_INFO ("Enable pcap tracing.");
87   //
88   // Do pcap tracing on all point-to-point devices on all nodes.
89   //
90   pointToPoint.EnablePcapAll ("star");
91
92
93
94 // Set the bounding box for animation
95   star.BoundingBox (1, 1, 100, 100);
96
97   // Create the animation object and configure for specified output
98   AnimationInterface anim ("star.xml");
99
100
101  NS_LOG_INFO ("Run Simulation.");
102  Simulator::Run ();
103  Simulator::Destroy ();
104  NS_LOG_INFO ("Done.");
105
106    return0    ;
107 }
```

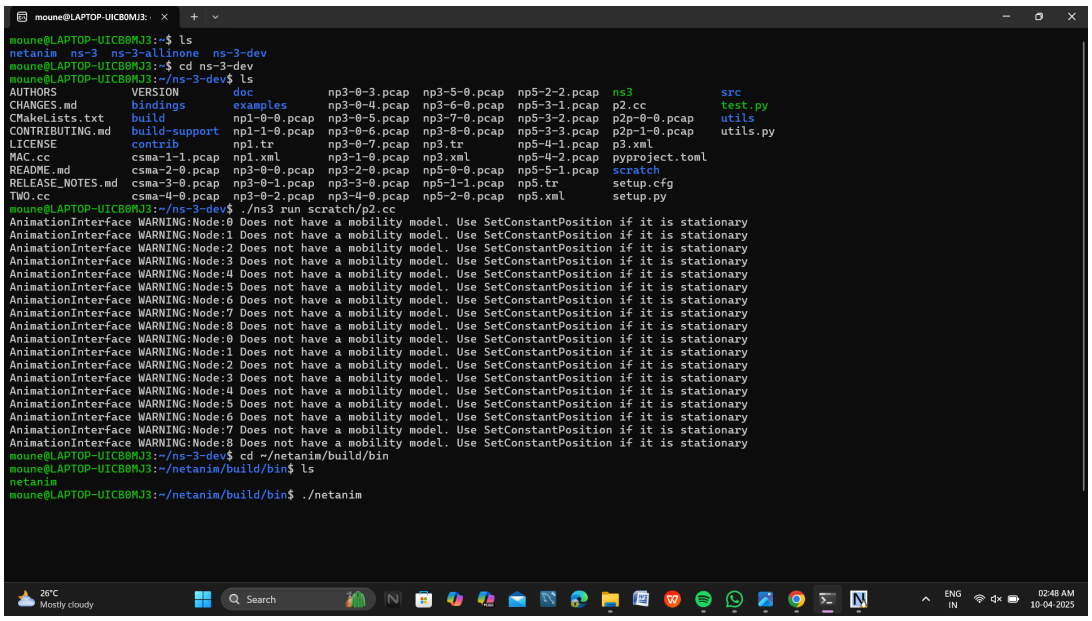
Listing 3: Star Topology in NS-3

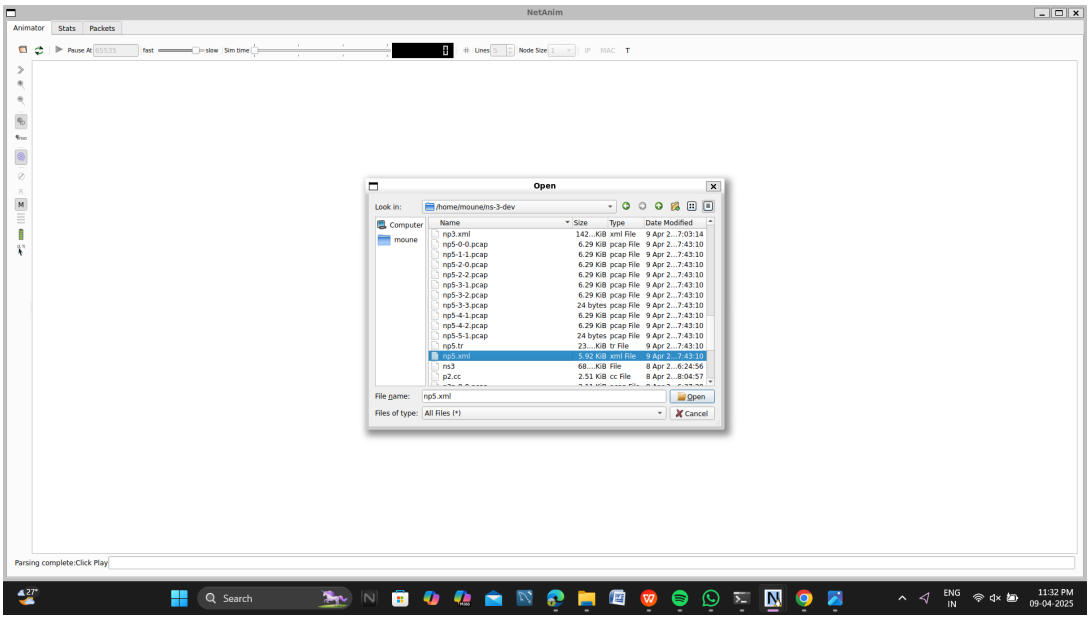
OUTPUT

In our system We have used these commands (Common for all programs)

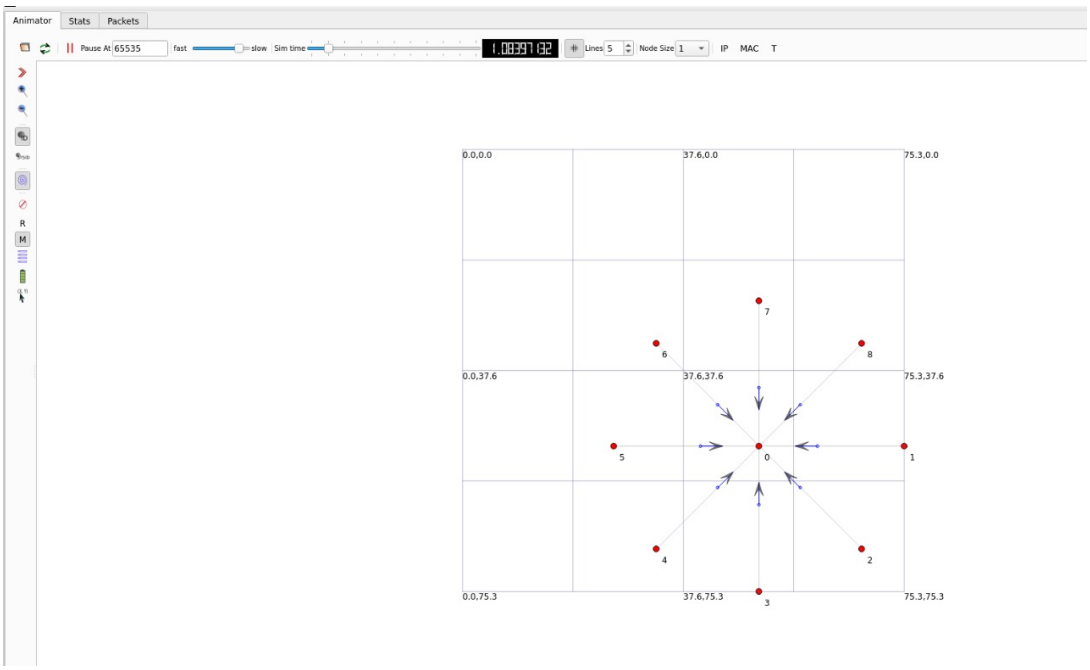
- **Open Ubuntu Terminal**
- `ls`
- `cd ns-3-dev` (File)
- `ls`
- **To create a file**
- `cd scratch`
- `nano filename.cc`
- Type a program
- `Ctrl+X` (To save)
- `Y + Enter`
- `cd ..`
- `./ns3 run scratch/filename.cc` (Run Created in scratch)
- *It will start building.*
- **If file not found, use 3 commands below:**
- `cd ns-3-dev`
- `cd build`
- `cmake ..`
- **Then: After running**
- `cd ~/netanim/build/bin`
- `ls`
- `./netanim`

OUTPUT


moune@LAPTOP-UICB0MJ3:~$ ls
netanim ns-3 ns-3-allinone ns-3-dev
moune@LAPTOP-UICB0MJ3:~$ cd ns-3-dev
moune@LAPTOP-UICB0MJ3:~/ns-3-dev$ ls
AUTHORS VERSION doc np3-0-3.pcap np3-5-0.pcap np5-2-2.pcap ns3 src
CHANGES.md bindings examples np3-0-4.pcap np3-6-0.pcap np5-3-1.pcap p2.cc test.py
CMakeLists.txt build np1-0-0.pcap np3-0-5.pcap np3-7-0.pcap np5-3-2.pcap p2p-0-0.pcap utils
CONTRIBUTING.md build-support np1-1-0.pcap np3-0-6.pcap np3-8-0.pcap np5-3-3.pcap p2p-1-0.pcap utils.py
LICENSE contrib np1.tr np3-0-7.pcap np3.tr np5-4-1.pcap p3.xml
MAC.cc csma-1-1.pcap np1.xml np3-1-0.pcap np3.xml np5-4-2.pcap pyproject.toml
README.md csma-2-0.pcap np3-0-0.pcap np3-2-0.pcap np5-0-0.pcap np5-5-1.pcap scratch
RELEASE_NOTES.md csma-3-0.pcap np3-0-1.pcap np3-3-0.pcap np5-1-1.pcap np5.tr setup.cfg
TWO.cc csma-4-0.pcap np3-0-2.pcap np3-4-0.pcap np5-2-0.pcap np5.xml setup.py
moune@LAPTOP-UICB0MJ3:~/ns-3-dev$ ./ns3 run scratch/p2.cc
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:5 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:6 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:7 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:8 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:5 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:6 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:7 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:8 Does not have a mobility model. Use SetConstantPosition if it is stationary
moune@LAPTOP-UICB0MJ3:~/ns-3-dev$ cd ~/netanim/build/bin
moune@LAPTOP-UICB0MJ3:~/netanim/build/bin$ ls
netanim
moune@LAPTOP-UICB0MJ3:~/netanim/build/bin$ ./netanim


NetAnim
Animator
Stats
Packets
Open
Look in:
/home/moune/ns-3-dev
Computer
moune
Name
Size
Type
Date Modified
np3.xml
np5-0-0.pcap
np5-1-1.pcap
np5-2-0.pcap
np5-2-2.pcap
np5-3-1.pcap
np5-3-2.pcap
np5-3-3.pcap
np5-4-1.pcap
np5-4-2.pcap
np5-5-1.pcap
np5.tr
np5.xml
ns3
p2.cc
File name:
np5.xml
Open
Files of type:
All Files (*)
Cancel
Parsing complete:Click Play
Search

OUTPUT


Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
Lines 5
Node Size 1
IP
MAC
T
0.0,0.0
37.6,0.0
75.3,0.0
0.0,37.6
37.6,37.6
75.3,37.6
0.0,75.3
37.6,75.3
75.3,75.3
0
1
2
3
4
5
6
7
8

12 Write a C++ program to implement a bus topology on NS-3

A bus topology connects all devices to a single communication line where each node can directly communicate with any other node on the bus. a bus topology using CSMA (Carrier Sense Multiple Access) in NS-3. It connects nodes using a point-to-point link and a shared CSMA channel.

Network Topology

In a bus topology:

- All nodes share a single channel or communication medium.
- Only one device can transmit at a time to avoid collisions.
- This is typically implemented using CSMA (Carrier Sense Multiple Access) in NS-3.

Simulation Details

Node Creation

Multiple nodes (e.g., five) are created to simulate devices connected to a shared medium. These nodes will communicate with each other using a shared channel.

Channel Configuration

A single `CsmaChannel` is created and shared among all nodes. The CSMA (Carrier Sense Multiple Access) protocol ensures that only one device transmits at a time.

Internet Stack and Addressing

Each node is installed with an Internet stack and given a unique IP address from the same subnet.

Routing

Since all nodes are on the same LAN (Local Area Network), no intermediate routers or routing configuration is required.

Applications

One node can act as a server, and another as a client. A UDP Echo Server is set up on one node, and a UDP Echo Client is set up on another. Communication occurs over the shared CSMA channel.

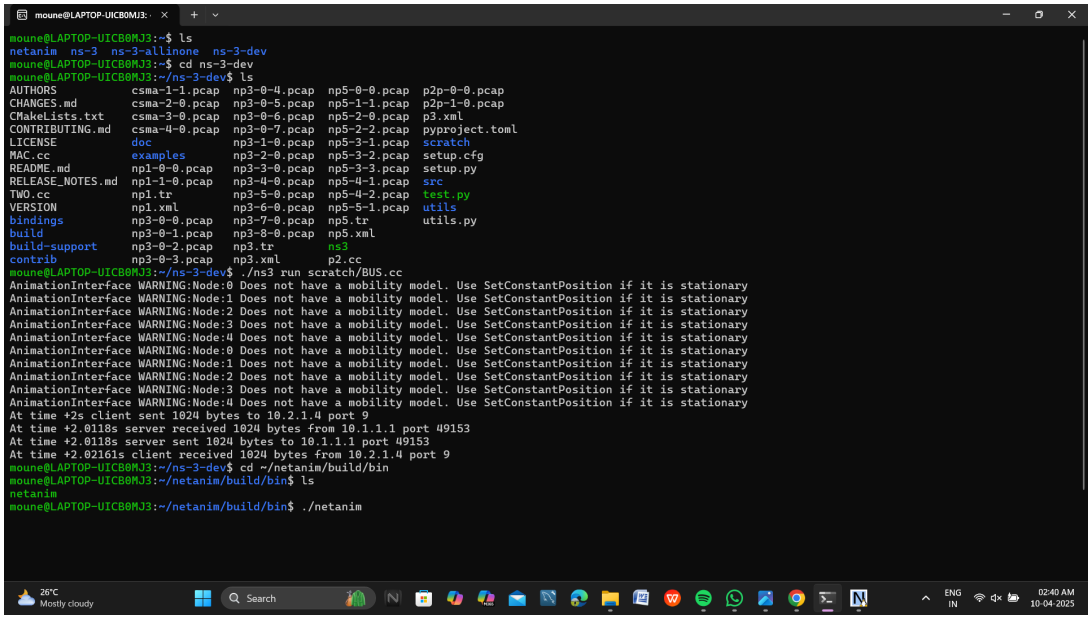
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"

// Default Network Bus Topology
//
//       10.1.1.0
// n0 -------------- n1   n2   n3   n4
//    point-to-point  |    |    |    |
//                    ================
//                      LAN 10.2.1.0-csma


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");
int
main (int argc, char *argv[])
{
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
 LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

  NodeContainer p2pNodes;
  p2pNodes.Create (2);

  NodeContainer csmaNodes;
  csmaNodes.Add (p2pNodes.Get (1));
  csmaNodes.Create (3);

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

  NetDeviceContainer p2pDevices;
  p2pDevices = pointToPoint.Install (p2pNodes);

  CsmaHelper csma;
  csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
  csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

  NetDeviceContainer csmaDevices;
  csmaDevices = csma.Install (csmaNodes);
```

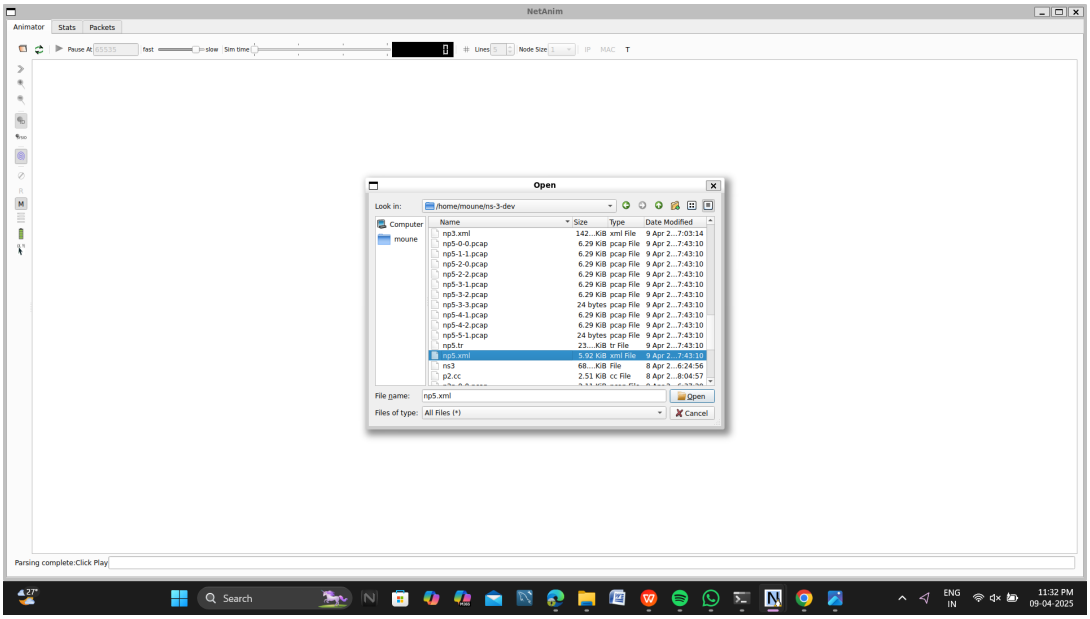
```
48
49   InternetStackHelper stack;
50   stack.Install (p2pNodes.Get(0));
51   stack.Install (csmaNodes);
52
53   Ipv4AddressHelper address;
54   address.SetBase ("10.1.1.0", "255.255.255.0");
55   Ipv4InterfaceContainer p2pInterfaces;
56   p2pInterfaces = address.Assign (p2pDevices);
57
58   address.SetBase ("10.2.1.0", "255.255.255.0");
59   Ipv4InterfaceContainer csmaInterfaces;
60   csmaInterfaces = address.Assign (csmaDevices);
61
62   UdpEchoServerHelper echoServer (9);
63
64   ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get
        (3));
65   serverApps.Start (Seconds (1.0));
66   serverApps.Stop (Seconds (10.0));
67
68   UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (3), 9);
69   echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
70   echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
71   echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
72
73   ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get
        (0));
74   clientApps.Start (Seconds (2.0));
75   clientApps.Stop (Seconds (10.0));
76
77   Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
78
79  pointToPoint.EnablePcapAll ("p2p");
80 csma.EnablePcapAll("csma");
81
82    AnimationInterface anim("p3.xml");
83     anim.SetConstantPosition(p2pNodes.Get(0),10.0,10.0);
84     anim.SetConstantPosition(csmaNodes.Get(0),20.0,20.0);
85     anim.SetConstantPosition(csmaNodes.Get(1),30.0,30.0);
86     anim.SetConstantPosition(csmaNodes.Get(2),40.0,40.0);
87     anim.SetConstantPosition(csmaNodes.Get(3),50.0,50.0);
88
89   Simulator::Run ();
90   Simulator::Destroy ();
91    return0    ;
```

```
92 }
```

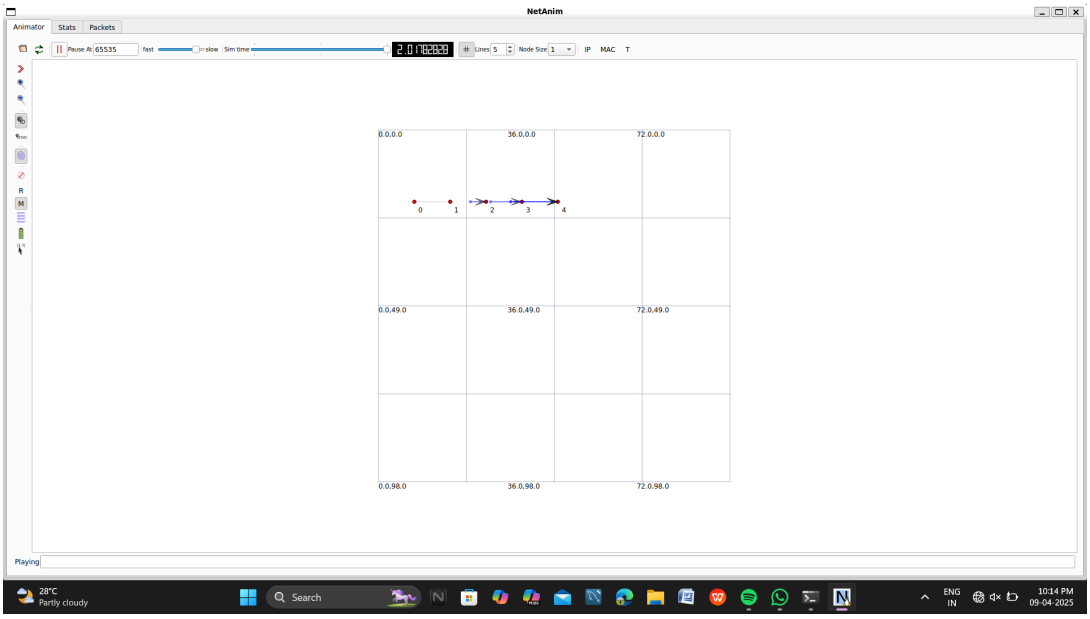
Listing 4: Bus Topology in NS-3

OUTPUT





OUTPUT


NetAnim
Animator
Stats
Packets
0.0.0.0
36.0.0.0
72.0.0.0
0.0.49.0
36.0.49.0
72.0.49.0
0.0.98.0
36.0.98.0
72.0.98.0
0
1
2
3
4
Playing
28°C
Partly cloudy
Search
ENG
IN
10:14 PM
09-04-2025

13 Write a C++ program showing the connection of two nodes and four routers such that the extreme nodes act as client and server on NS-3

Where two nodes are connected through four routers. The nodes at the ends act as client and server, respectively. where two end nodes are connected through four intermediate routers using point-to-point links. The two end nodes act as a client and a server respectively, communicating over a multihop network.

Network Topology

The network consists of:

- Two end nodes: Node0 (Client) and Node1 (Server)
- Four intermediate routers: R1, R2, R3, and R4
- Five point-to-point links connecting the entire path

The structure is illustrated as:

$$\text{Node0 (Client)} \rightarrow \text{R1} \rightarrow \text{R2} \rightarrow \text{R3} \rightarrow \text{R4} \rightarrow \text{Node1 (Server)}$$

Each link is defined with a specified bandwidth and delay to emulate real network conditions.

Simulation Details

Node and Router Creation

Two end nodes are created to represent the client and server. Four additional nodes are created to serve as routers. This totals six nodes.

Link Configuration

Five point-to-point connections are configured between:

1. Node0 and Router1
2. Router1 and Router2
3. Router2 and Router3
4. Router3 and Router4
5. Router4 and Node1

Each link is configured with a data rate of `5Mbps` and a delay of `2ms`.

Internet Stack and Addressing

The Internet stack is installed on all nodes, and each link is assigned a unique IP subnet. This allows routing of packets across the network.

Routing

Global routing is used to automatically populate routing tables across all nodes. This ensures the client can reach the server through the intermediate routers.

Applications

A UDP Echo Server is installed on Node1 (Server) and a UDP Echo Client is installed on Node0 (Client). The client sends a number of packets to the server and receives replies, validating the multihop connection.

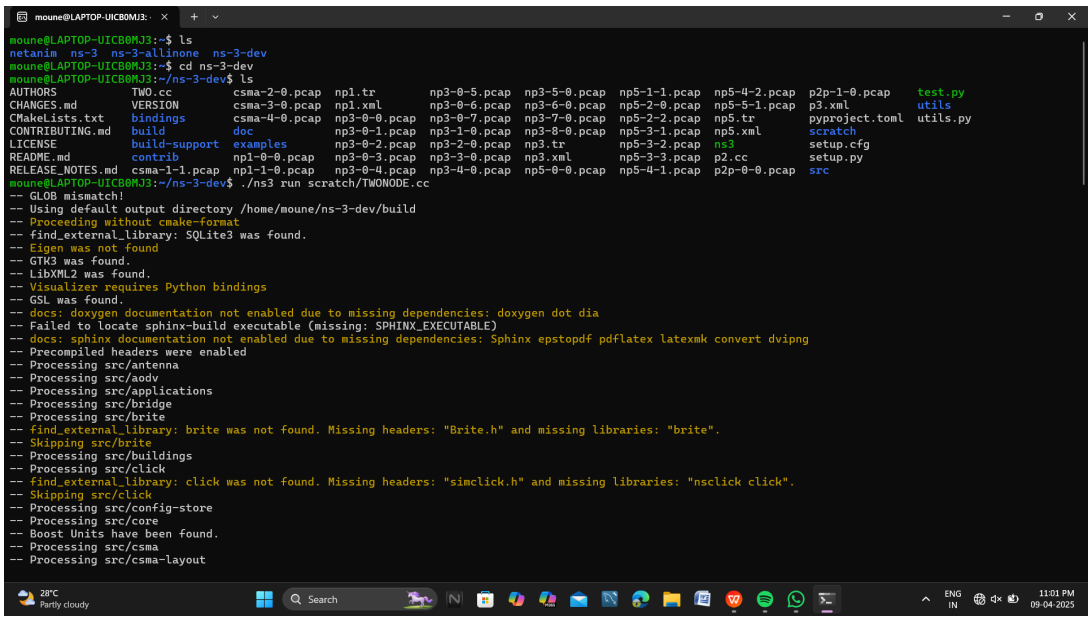
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");

int main (int argc, char *argv[])
{
    bool verbose = true;

    if (verbose)
    {
        LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
        LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }

    NodeContainer host, router;
    host.Create (2);        // Two end-hosts: client and server
    router.Create (4);      // Four intermediate routers

    // Subnet 1: Host 0 to Router 0
    NodeContainer subnet1;
    subnet1.Add (host.Get (0));
    subnet1.Add (router.Get (0));

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

    NetDeviceContainer subnet1Devices = pointToPoint.Install (subnet1);

    InternetStackHelper stack;
    stack.Install (router);
    stack.Install (host);

    Ipv4AddressHelper address1;
    address1.SetBase ("10.1.1.0", "255.255.255.0");
```

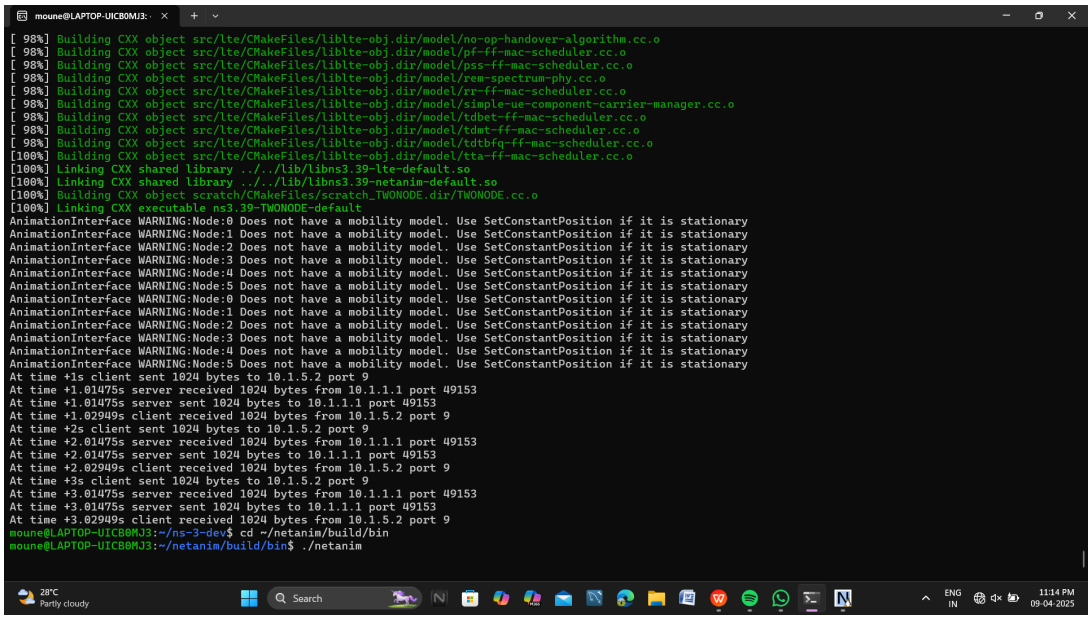
```
45      Ipv4InterfaceContainer subnet1Interfaces = address1.Assign (
            subnet1Devices);
46
47      // Subnet 2: Router 0 to Router 1
48      NodeContainer subnet2;
49      subnet2.Add (router.Get (0));
50      subnet2.Add (router.Get (1));
51      NetDeviceContainer subnet2Devices = pointToPoint.Install (subnet2);
52      Ipv4AddressHelper address2;
53      address2.SetBase ("10.1.2.0", "255.255.255.0");
54      Ipv4InterfaceContainer subnet2Interfaces = address2.Assign (
            subnet2Devices);
55
56      // Subnet 3: Router 1 to Router 2
57      NodeContainer subnet3;
58      subnet3.Add (router.Get (1));
59      subnet3.Add (router.Get (2));
60      NetDeviceContainer subnet3Devices = pointToPoint.Install (subnet3);
61      Ipv4AddressHelper address3;
62      address3.SetBase ("10.1.3.0", "255.255.255.0");
63      Ipv4InterfaceContainer subnet3Interfaces = address3.Assign (
            subnet3Devices);
64
65      // Subnet 4: Router 1 to Router 3
66      NodeContainer subnet4;
67      subnet4.Add (router.Get (1));
68      subnet4.Add (router.Get (3));
69      NetDeviceContainer subnet4Devices = pointToPoint.Install (subnet4);
70      Ipv4AddressHelper address4;
71      address4.SetBase ("10.1.4.0", "255.255.255.0");
72      Ipv4InterfaceContainer subnet4Interfaces = address4.Assign (
            subnet4Devices);
73
74      // Subnet 5: Router 2 to Host 1
75      NodeContainer subnet5;
76      subnet5.Add (router.Get (2));
77      subnet5.Add (host.Get (1));
78      NetDeviceContainer subnet5Devices = pointToPoint.Install (subnet5);
79      Ipv4AddressHelper address5;
80      address5.SetBase ("10.1.5.0", "255.255.255.0");
81      Ipv4InterfaceContainer subnet5Interfaces = address5.Assign (
            subnet5Devices);
82
83      // Server Application on Host 1
84      UdpEchoServerHelper echoServer (9);
85      ApplicationContainer serverApps = echoServer.Install (subnet5.Get
            (1));
```

```
86      serverApps.Start (Seconds (1.0));
87      serverApps.Stop (Seconds (10.0));
88
89      // Client Application on Host 0
90      UdpEchoClientHelper echoClient (subnet5Interfaces.GetAddress (1),
            9);
91      echoClient.SetAttribute ("MaxPackets", UintegerValue (3));
92      echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
93      echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
94      ApplicationContainer clientApps = echoClient.Install (subnet1.Get
            (0));
95      clientApps.Start (Seconds (1.0));
96      clientApps.Stop (Seconds (10.0));
97
98      Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
99
100     AnimationInterface anim("np5.xml");
101
102     AsciiTraceHelper eventTraces;
103     pointToPoint.EnableAsciiAll(eventTraces.CreateFileStream("np5.tr"))
            ;
104     pointToPoint.EnablePcapAll ("np5");
105
106     Simulator::Run ();
107     Simulator::Destroy ();
108
109     return 0;
110 }
```

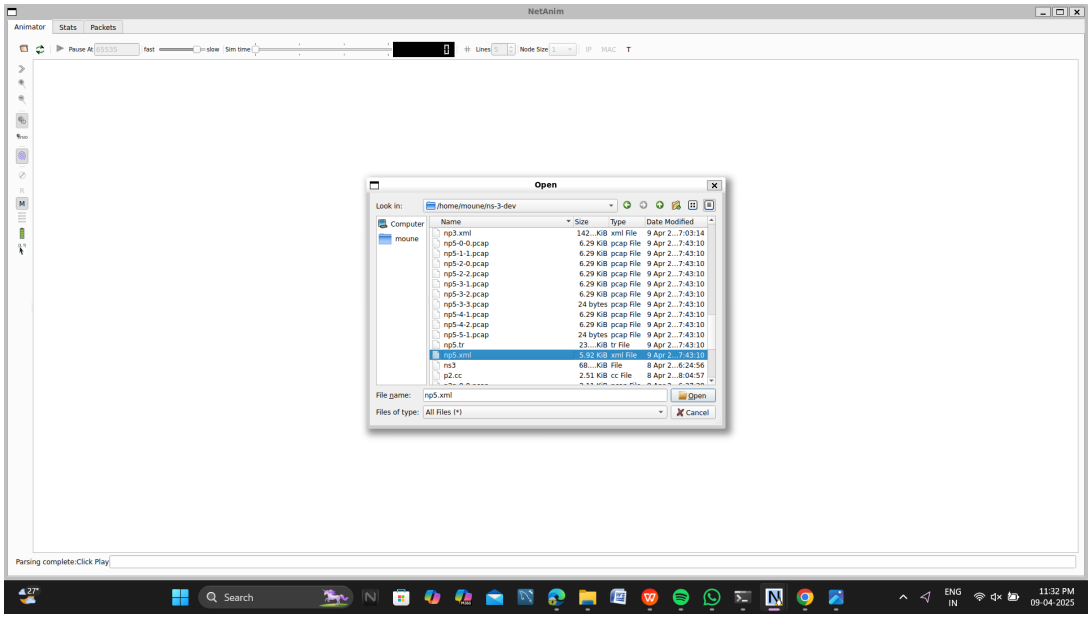
Listing 5: Two Nodes and Four Routers in NS-3

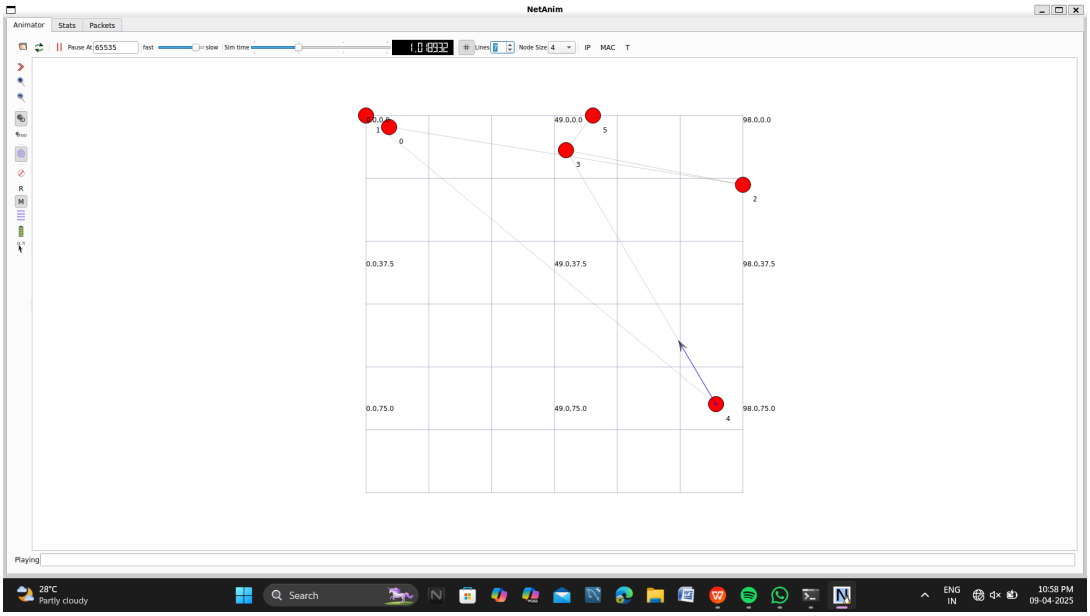
OUTPUT


moune@LAPTOP-UICB0MJ3:~$ ls
netanim ns-3 ns-3-allinone ns-3-dev
moune@LAPTOP-UICB0MJ3:~$ cd ns-3-dev
moune@LAPTOP-UICB0MJ3:~/ns-3-dev$ ls
AUTHORS TWO.cc csma-2-0.pcap np1.tr np3-0-5.pcap np3-5-0.pcap np5-1-1.pcap np5-4-2.pcap p2p-1-0.pcap test.py
CHANGES.md VERSION csma-3-0.pcap np1.xml np3-0-6.pcap np3-6-0.pcap np5-2-0.pcap np5-5-1.pcap p3.xml utils
CMakeLists.txt bindings csma-4-0.pcap np3-0-0.pcap np3-0-7.pcap np3-7-0.pcap np5-2-2.pcap np5.tr pyproject.toml utils.py
CONTRIBUTING.md build doc np3-0-1.pcap np3-1-0.pcap np3-8-0.pcap np5-3-1.pcap np5.xml scratch
LICENSE build-support examples np3-0-2.pcap np3-2-0.pcap np3.tr np5-3-2.pcap ns3 setup.cfg
README.md contrib np1-0-0.pcap np3-0-3.pcap np3-3-0.pcap np3.xml np5-3-3.pcap p2.cc setup.py
RELEASE_NOTES.md csma-1-1.pcap np1-1-0.pcap np3-0-4.pcap np3-4-0.pcap np5-0-0.pcap np5-4-1.pcap p2p-0-0.pcap src
moune@LAPTOP-UICB0MJ3:~/ns-3-dev$ ./ns3 run scratch/TWONODE.cc
-- GLOB mismatch!
-- Using default output directory /home/moune/ns-3-dev/build
-- Proceeding without cmake-format
-- find_external_library: SQLite3 was found.
-- Eigen was not found
-- GTK3 was found.
-- LibXML2 was found.
-- Visualizer requires Python bindings
-- GSL was found.
-- docs: doxygen documentation not enabled due to missing dependencies: doxygen dot dia
-- Failed to locate sphinx-build executable (missing: SPHINX_EXECUTABLE)
-- docs: sphinx documentation not enabled due to missing dependencies: Sphinx epstopdf pdflatex latexmk convert dvipng
-- Precompiled headers were enabled
-- Processing src/antenna
-- Processing src/aodv
-- Processing src/applications
-- Processing src/bridge
-- Processing src/brite
-- find_external_library: brite was not found. Missing headers: "Brite.h" and missing libraries: "brite".
-- Skipping src/brite
-- Processing src/buildings
-- Processing src/click
-- find_external_library: click was not found. Missing headers: "simclick.h" and missing libraries: "nsclick click".
-- Skipping src/click
-- Processing src/config-store
-- Processing src/core
-- Boost Units have been found.
-- Processing src/csma
-- Processing src/csma-layout


[98%] Building CXX object src/lte/CMakeFiles/liblte-obj.dir/model/no-op-handover-algorithm.cc.o
[98%] Building CXX object src/lte/CMakeFiles/liblte-obj.dir/model/pf-ff-mac-scheduler.cc.o
[98%] Building CXX object src/lte/CMakeFiles/liblte-obj.dir/model/pss-ff-mac-scheduler.cc.o
[98%] Building CXX object src/lte/CMakeFiles/liblte-obj.dir/model/rem-spectrum-phy.cc.o
[98%] Building CXX object src/lte/CMakeFiles/liblte-obj.dir/model/rr-ff-mac-scheduler.cc.o
[98%] Building CXX object src/lte/CMakeFiles/liblte-obj.dir/model/simple-ue-component-carrier-manager.cc.o
[98%] Building CXX object src/lte/CMakeFiles/liblte-obj.dir/model/tdbet-ff-mac-scheduler.cc.o
[98%] Building CXX object src/lte/CMakeFiles/liblte-obj.dir/model/tdmt-ff-mac-scheduler.cc.o
[98%] Building CXX object src/lte/CMakeFiles/liblte-obj.dir/model/tdtbfq-ff-mac-scheduler.cc.o
[100%] Building CXX object src/lte/CMakeFiles/liblte-obj.dir/model/tta-ff-mac-scheduler.cc.o
[100%] Linking CXX shared library ../../lib/libns3.39-lte-default.so
[100%] Linking CXX shared library ../../lib/libns3.39-netanim-default.so
[100%] Building CXX object scratch/CMakeFiles/scratch_TWONODE.dir/TWONODE.cc.o
[100%] Linking CXX executable ns3.39-TWONODE-default
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:5 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:5 Does not have a mobility model. Use SetConstantPosition if it is stationary
At time +1s client sent 1024 bytes to 10.1.5.2 port 9
At time +1.01475s server received 1024 bytes from 10.1.1.1 port 49153
At time +1.01475s server sent 1024 bytes to 10.1.1.1 port 49153
At time +1.02949s client received 1024 bytes from 10.1.5.2 port 9
At time +2s client sent 1024 bytes to 10.1.5.2 port 9
At time +2.01475s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.01475s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.02949s client received 1024 bytes from 10.1.5.2 port 9
At time +3s client sent 1024 bytes to 10.1.5.2 port 9
At time +3.01475s server received 1024 bytes from 10.1.1.1 port 49153
At time +3.01475s server sent 1024 bytes to 10.1.1.1 port 49153
At time +3.02949s client received 1024 bytes from 10.1.5.2 port 9
moune@LAPTOP-UICB0MJ3:~/ns-3-dev$ cd ~/netanim/build/bin
moune@LAPTOP-UICB0MJ3:~/netanim/build/bin$ ./netanim

OUTPUT


NetAnim
Animator
Stats
Packets
Open
Look in:
/home/moune/ns-3-dev
Computer
moune
Name
Size
Type
Date Modified
np5.xml
File name:
Files of type:
All Files (*)
Open
Cancel
Parsing complete:Click Play
Search
ENG
IN
11:32 PM
09-04-2025


NetAnim
Animator
Stats
Packets
49.0.0.0
98.0.0.0
0.0.37.5
49.0.37.5
98.0.37.5
0.0.75.0
49.0.75.0
98.0.75.0
Playing
Search
ENG
IN
10:58 PM
09-04-2025

14 Implement and study the performance of a typical GSM network on NS-3 (using MAC layer).

The Global System for Mobile Communications (GSM) is a standard developed to describe protocols for second generation (2G) digital cellular networks. While NS-3 (Network Simulator 3) does not include a full GSM model, it can be adapted to simulate basic GSM-like behaviors at the MAC layer using CSMA or LTE primitives.

This project aims to implement a simplified GSM-like network using the CSMA protocol to emulate MAC-layer communication, and study the performance of such a system in terms of throughput, latency, and packet delivery.

NS-3

NS-3 is a discrete-event network simulator, popular in academic and research settings. It supports network models for internet stacks, routing, WiFi, LTE, and more.

GSM

GSM architecture consists of mobile devices, base transceiver stations (BTS), and a core network. The MAC layer in GSM handles time slot scheduling, retransmissions, and control signaling.

Simulation Design

Topology

- 1 Base Station (BTS)
- 3 User Equipment (UE) nodes
- CSMA channel for communication

Assumptions

- GSM behavior is emulated via CSMA MAC channel
- UDP echo application simulates voice or message transfer
- Basic metrics like delay and throughput are analyzed

Implementation in C++ (NS-3)

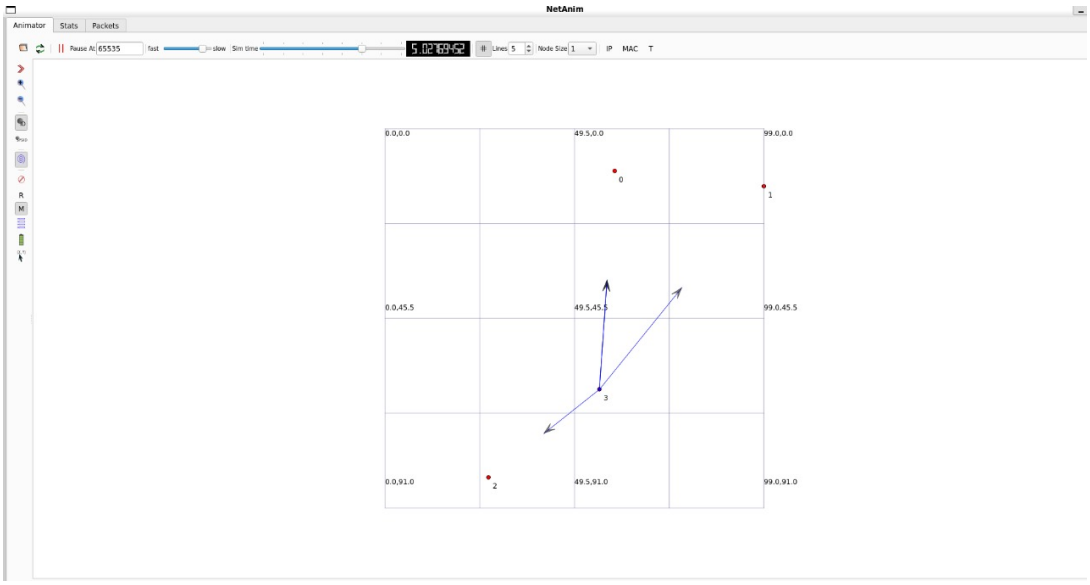
Simulation Code

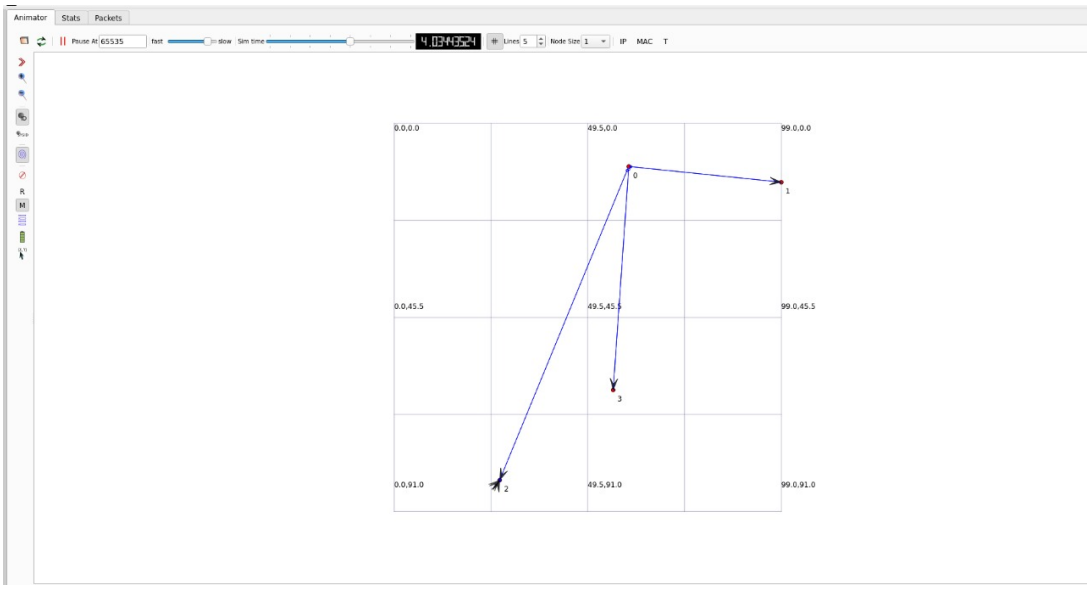
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include "ns3/netanim-module.h"
using namespace ns3;

NS_LOG_COMPONENT_DEFINE("GsmMacSim");

int main(int argc, char *argv[]) {
    NodeContainer nodes;
    nodes.Create(4);

    CsmaHelper csma;
    csma.SetChannelAttribute("DataRate", StringValue("1Mbps"));
    csma.SetChannelAttribute("Delay", TimeValue(NanoSeconds(6560)
       ));

    NetDeviceContainer devices = csma.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(0));
    serverApps.Start(Seconds(1.0));
    serverApps.Stop(Seconds(10.0));

    for (int i = 1; i < 4; ++i) {
```

```
35          UdpEchoClientHelper echoClient(interfaces.GetAddress(0),
               9);
36          echoClient.SetAttribute("MaxPackets", UintegerValue(3));
37          echoClient.SetAttribute("Interval", TimeValue(Seconds
               (1.0)));
38          echoClient.SetAttribute("PacketSize", UintegerValue(1024)
               );
39
40          ApplicationContainer clientApps = echoClient.Install(
               nodes.Get(i));
41          clientApps.Start(Seconds(2.0 + i));
42          clientApps.Stop(Seconds(10.0));
43      }
44
45      csma.EnablePcapAll("gsm-mac");
46      AnimationInterface anim("gm.xml");
47      Simulator::Run();
48      Simulator::Destroy();
49        return0       ;
50  }
```

Listing 6: Simplified GSM MAC Simulation

OUTPUT


NetAnim
Animator
Stats
Packets
0.0,0.0
49.5,0.0
99.0,0.0
0.0,45.5
49.5,45.5
99.0,45.5
0.0,91.0
49.5,91.0
99.0,91.0
0
1
2
3


Animator
Stats
Packets
0.0,0.0
49.5,0.0
99.0,0.0
0.0,45.5
49.5,45.5
99.0,45.5
0.0,91.0
49.5,91.0
99.0,91.0
0
1
2
3