**TAGORE ENGINEERING COLLEGE**

## RATHINAMANGALAM, CHENNAI-127.



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

CS3591 – COMPUTER NETWORKS

# Name :

**Reg. No :**

# Branch :

**Year :**

# Semester :

## Anna University∷Chennai Regulation 2021

**TAGORE ENGINEERING COLLEGE**

**RATHINAMANGALAM, CHENNAI-600127**

# LABORATORY RECORD

UNIVERSITY REGISTER NUMBER:

Certified that this is the bonafide record of work done by Mr./Ms. of

Department in the laboratory and submitted for the University Practical Examination conducted on

at TAGORE ENGINEERING COLLEGE, CHENNAI-127.

Record Marks:

## Lab In-charge Principal Head of the Department

**External Examiner Internal Examiner**

**VISION:**

To create an environment which is conducive to produce competent Computer Science Engineers through quality education and research-oriented education and equip them for the needs of the industry and society.

**MISSION:**

The Department strives to contribute to the expansion of knowledge in the discipline of Computer Science and Engineering by

* Adopting an efficient teaching learning process in concurrence with increasing industrial demands.
* Ensuring technical proficiency, facilitating to pursue higher studies and carry out Research & Development activities.
* Developing problem solving and analytical skills with deep knowledge in thorough understanding of basic sciences and Computer Science Engineering.
* Infusing managerial and entrepreneurship skills to become ethical, socially responsible and competitive professionals.

**PROGRAM SPECIFIC OUTCOMES:**

* Design and development of software and firmware solutions using latest Computer Science tools and technologies to address societal problems.
* Apply acquired knowledge to involve enthusiastically in software development, software testing, storage, computing and business intelligence sectors.
* Create a conducive environment to excel in their career by using their technical expertise in the latest technologies and update their knowledge continuously in Computer Science and Engineering.

**PROGRAMME OUTCOMES:**

Engineering Graduates will be able to:

* Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
* Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
* Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
* Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
* Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
* The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
* Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
* Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
* Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
* Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
* Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
* Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**COURSE OUTCOMES**

* **CO1:** Implement various protocols using TCP and UDP
* **CO2:** Compare the performance of different transport layer protocols.
* **CO3:** Use simulation tools to analyze the performance of various network protocols
* **CO4:** Analyze various routing algorithms
* **CO5:** Implement error correction codes.

**TABLE OF CONTENTS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Exp.No.** | **Date** | **Title of the Experiment** | **Page No.** | **Marks** | **Signature** |
| **1** |  | **Learn to use commands like tcpdump, netstat, ifconfig, nslookup, and traceroute. Capture ping and traceroute pdus using a network protocol analyzer and examine** |  |  |  |
| **2** |  | **Implementation of an HTTP web client program to download a web page using TCP sockets** |  |  |  |
| **3** |  | **Developing TCP socket applications: echo client and server, and chat** |  |  |  |
| **4** |  | **Simulation of DNS using UDP sockets** |  |  |  |
| **5** |  | **Installing and configuring wireshark and capturing/examining network packets** |  |  |  |
| **6** |  | **Simulation of ARP /RARP protocols** |  |  |  |
| **7** |  | **Study of Network Simulator (NS) and Simulation of Congestion Control Algorithms using NS** |  |  |  |
| **8** |  | **Study of TCP/UDP performance using simulation tool** |  |  |  |
| **9** |  | **Simulation of distance vector/ link state routing algorithm** |  |  |  |
| **10** |  | **Simulation of Error Correction Code (CRC)** |  |  |  |

**DATE:**

**EXP.NO.:**01

**LEARN TO USE COMMANDS LIKE TCPDUMP, NETSTAT, IFCONFIG, NSLOOKUP, AND TRACEROUTE. CAPTURE PING AND TRACEROUTE PDUS USING A NETWORK PROTOCOL ANALYZER AND EXAMINE.**

**AIM:**

To learn how to use commands such as tcpdump, netstat, ifconfig, nslookup, traceroute, and ping, and to capture and examine ping and traceroute PDUs (Protocol Data Units) using a network protocol analyzer.

**PRE-LAB DISCUSSION:**

**TCPDUMP:**

The tcpdump utility allows you to capture packets that flow within your network, assisting in network troubleshooting. Here are several examples of using tcpdump with different options. Traffic is captured based on a specified filter.

**NETSTAT:**

Netstat is a common command-line TCP/IP networking tool available in most versions of Windows, Linux, UNIX, and other operating systems. Netstat provides information and statistics about protocols in use and current TCP/IP network connections.

**IFCONFIG (LINUX) OR IPCONFIG (WINDOWS):**

Ifconfig is a command used in Linux, while ipconfig is a Windows utility. These tools allow you to retrieve IP address information for a computer. To use ifconfig in Linux, you would typically run it from the terminal. In Windows, open the command prompt and type ipconfig to obtain IP address, network mask, and gateway information for all physical and virtual network adapters.

**NSLOOKUP:**

Nslookup (short for name server lookup) is a network utility program used to obtain information about internet servers. It is employed to find name server information for domains by querying the Domain Name System.

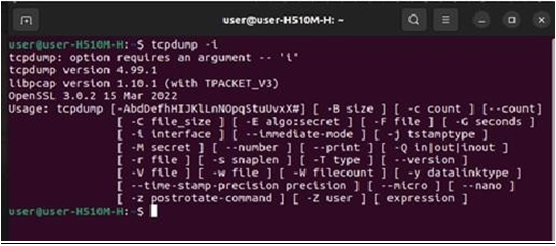
**TRACEROUTE:**

Traceroute is a network diagnostic tool used to trace the path taken by a packet on an IP network from source to destination. Traceroute also records the time taken for each hop the packet makes during its route to the destination.

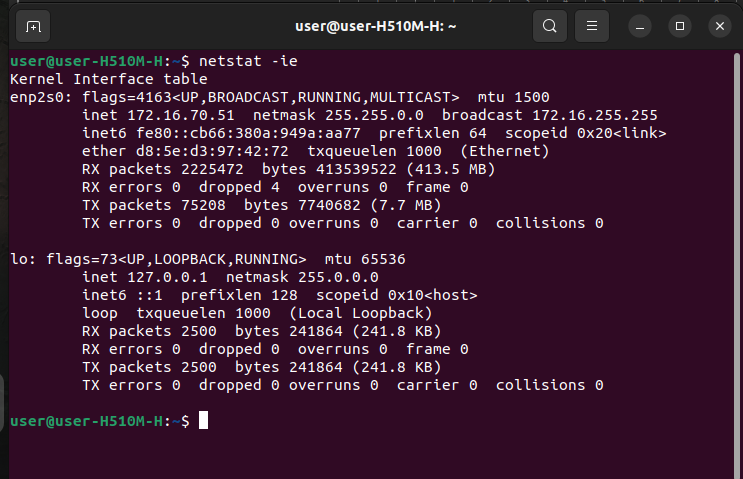
**COMMANDS:**

**TCPDUMP:**

Tcpdump is a powerful utility that allows you to capture and analyze network packets within your network, which is essential for network troubleshooting. You can use various options and filters to capture specific network traffic for analysis.

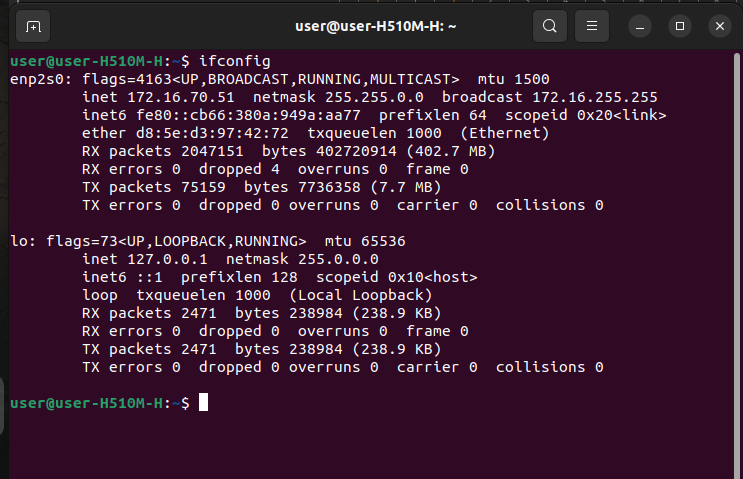


**NETSTAT:**

Netstat is a common command-line tool available in most versions of Windows, Linux, UNIX, and other operating systems. It provides valuable information and statistics about the protocols in use and the current TCP/IP network connections. On Windows, running `netstat` displays protocol statistics and current network connections.

**IPCONFIG (WINDOWS) OR IFCONFIG (LINUX):**

Ipconfig is a console application in Windows that is used to obtain IP address information for a Windows computer. In contrast, ifconfig is used in Linux to configure and display network interface parameters. To obtain IP address, network mask, and gateway information in Windows, you can run `ipconfig` in the command prompt. In Linux, you can use `ifconfig` to configure and display network settings.



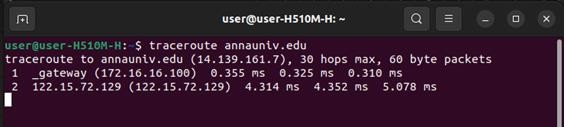
**NSLOOKUP:**

Nslookup, short for "name server lookup," is a network utility program used to obtain information about internet servers. It can be used to find name server information for domains by querying the Domain Name System (DNS). Nslookup is a valuable tool for diagnosing DNS-related issues and is used when you can access a resource by specifying its IP address but not its DNS name.



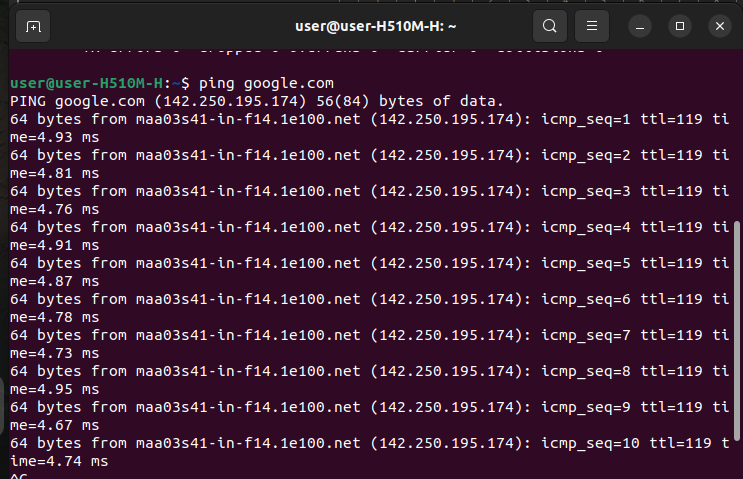
**TRACEROUTE:**

Traceroute is a network diagnostic tool used to trace the path taken by a packet on an IP network from its source to its destination. It records the time taken for each hop the packet makes during its route to the destination. Traceroute utilizes Internet Control Message Protocol (ICMP) echo packets with variable time to live (TTL) values. The response time for each hop is calculated, and routers are queried multiple times to measure their response accurately. The traceroute command sends packets with TTL values that gradually increase, starting with a TTL value of one, and builds a list of routers that packets traverse until the destination is reached.



**PING:**

The ping command sends an echo request to a host available on the network to check if it's responding. It is a basic tool for tracking and isolating hardware and software problems and determining the network and host status. The ping command operates by sending Internet Control Message Protocol (ICMP) Echo Request messages to the destination computer and waiting for a response. It is commonly used to verify network connectivity and can help diagnose network issues. For example, you can use the command `ping www.google.com` to check if you can reach the Google website.



**RESULT:**

Thus, various network commands such as tcpdump, netstat, ifconfig, nslookup, traceroute, and ping have been executed successfully.

**PROGRAM:**

**Client (HTTPClient.java):**

import javax.swing.\*;

import java.net.\*;

import java.awt.image.\*;

import javax.imageio.\*;

import java.io.\*;

import java.awt.image.BufferedImage;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;

public class HTTPClient {

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

Socket soc;

BufferedImage img = null;

soc = new Socket("localhost", 9999);

System.out.println("Client is running.");

try {

System.out.println("Reading image from disk.");

img = ImageIO.read(new File("car-wallpaper-3.jpg"));

ByteArrayOutputStream baos = new ByteArrayOutputStream();

ImageIO.write(img, "jpg", baos);

baos.flush();

byte[] bytes = baos.toByteArray();

baos.close();

System.out.println("Sending image to the server.");

OutputStream out = soc.getOutputStream();

DataOutputStream dos = new DataOutputStream(out);

dos.writeInt(bytes.length);

dos.write(bytes, 0, bytes.length);

System.out.println("Image sent to the server.");

} catch (Exception e) {

System.out.println("Exception: " + e.getMessage());

soc.close();

}

soc.close();

}

}

**Server (HTTPServer.java):**

import java.net.\*;

import java.io.\*;

import java.awt.image.\*;

import javax.imageio.\*;

import javax.swing.\*;

class HTTPServer {

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

ServerSocket server = null;

Socket socket;

server = new ServerSocket(9999);

System.out.println("Server Waiting for an image.");

socket = server.accept();

System.out.println("Client connected.");

InputStream in = socket.getInputStream();

DataInputStream dis = new DataInputStream(in);

int len = dis.readInt();

System.out.println("Image Size: " + len / 1024 + "KB");

byte[] data = new byte[len];

dis.readFully(data);

dis.close();

in.close();

InputStream ian = new ByteArrayInputStream(data);

BufferedImage bImage = ImageIO.read(ian);

JFrame f = new JFrame("Server");

ImageIcon icon = new ImageIcon(bImage);

JLabel l = new JLabel();

l.setIcon(icon);

f.add(l);

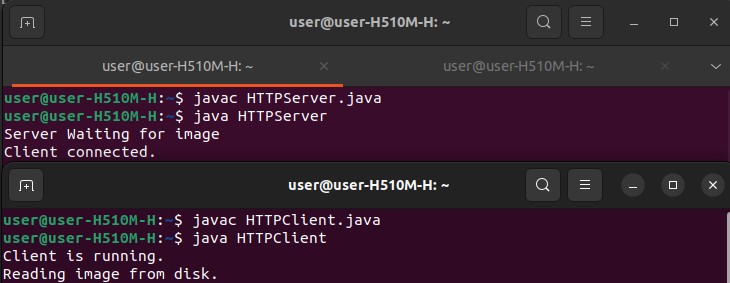
f.pack();

f.setVisible(true);

}

}

**OUTPUT:**



**PROGRAM:**

**EchoClient.java:**

import java.io.\*;

import java.net.\*;

public class EchoClient {

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

Socket soc = null;

PrintWriter out = null;

BufferedReader in = null;

try {

soc = new Socket("localhost", 9999);

out = new PrintWriter(soc.getOutputStream(), true);

in = new BufferedReader(new InputStreamReader(soc.getInputStream()));

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String msg;

while (true) {

System.out.print("Enter a message (or type 'exit' to quit): ");

msg = br.readLine();

if (msg.equals("exit")) {

break; // Exit the loop and close the client

}

out.println(msg);

String response = in.readLine();

System.out.println("Echo: " + response);

}

} catch (IOException e) {

e.printStackTrace();

} finally {

if (soc != null) {

soc.close();

}

}

}

}

**EchoServer.java:**

import java.io.\*;

import java.net.\*;

public class EchoServer {

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

ServerSocket server = null;

Socket clientSocket = null;

PrintWriter out = null;

BufferedReader in = null;

try {

server = new ServerSocket(9999);

System.out.println("Server is waiting for connections...");

clientSocket = server.accept();

out = new PrintWriter(clientSocket.getOutputStream(), true);

in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

String msg;

while ((msg = in.readLine()) != null) {

System.out.println("Received: " + msg);

out.println(msg); // Echo back to the client

}

} catch (IOException e) {

e.printStackTrace();

} finally {

if (server != null) {

server.close();

}

if (clientSocket != null) {

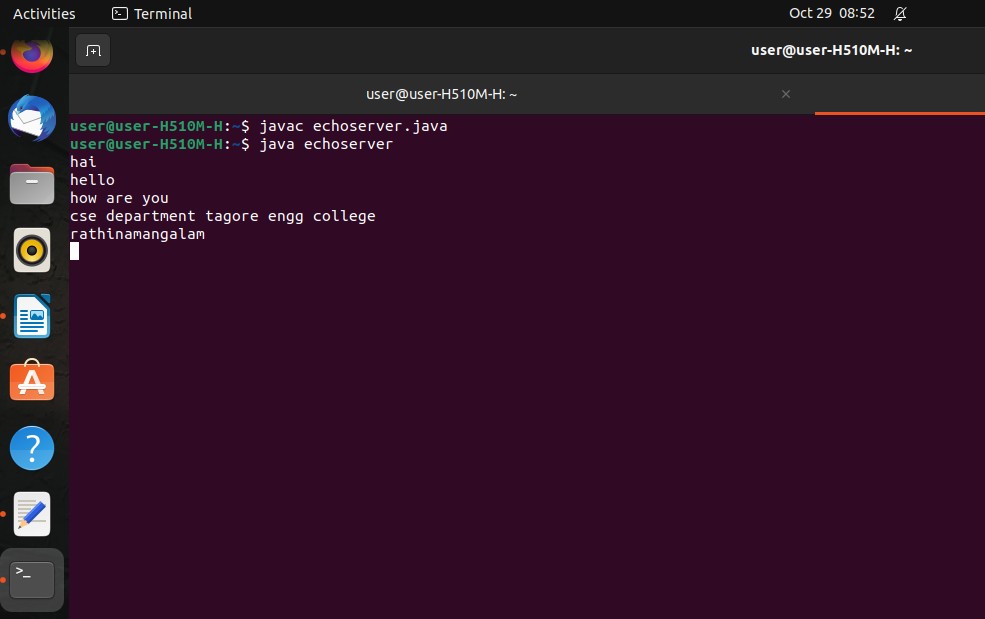
clientSocket.close();

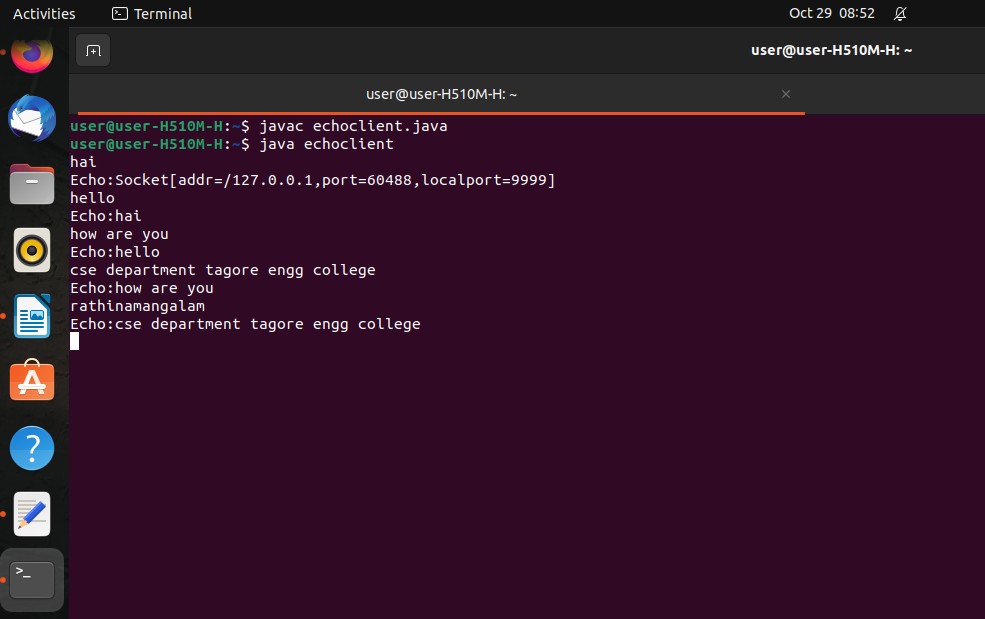
}

}

}

}

**OUTPUT:**



**PROGRAM:**

**TCPClient.java:**

import java.net.\*;

import java.io.\*;

public class TCPClient {

public static void main(String[] args) {

Socket c = null;

String line;

DataInputStream is, is1;

PrintStream os;

try {

c = new Socket("localhost", 9999); // Change the IP address to "localhost" or the server's IP

} catch (IOException e) {

System.out.println(e);

}

try {

os = new PrintStream(c.getOutputStream());

is = new DataInputStream(System.in);

is1 = new DataInputStream(c.getInputStream());

do {

System.out.println("Client:");

line = is.readLine();

os.println(line);

System.out.println("Server: " + is1.readLine());

} while (!line.equalsIgnoreCase("quit"));

is1.close();

os.close();

} catch (IOException e) {

System.out.println("Socket Closed! Message Passing is over");

}

}

}

**TCPServer.java:**

import java.net.\*;

import java.io.\*;

public class TCPServer {

public static void main(String[] args) {

ServerSocket s = null;

String line;

DataInputStream is, is1;

PrintStream os = null;

Socket c = null;

try {

s = new ServerSocket(9999);

} catch (IOException e) {

System.out.println(e);

}

try {

c = s.accept();

is = new DataInputStream(c.getInputStream());

is1 = new DataInputStream(System.in);

os = new PrintStream(c.getOutputStream());

do {

line = is.readLine();

System.out.println("Client: " + line);

System.out.println("Server:");

line = is1.readLine();

os.println(line);

} while (!line.equalsIgnoreCase("quit"));

is.close();

os.close();

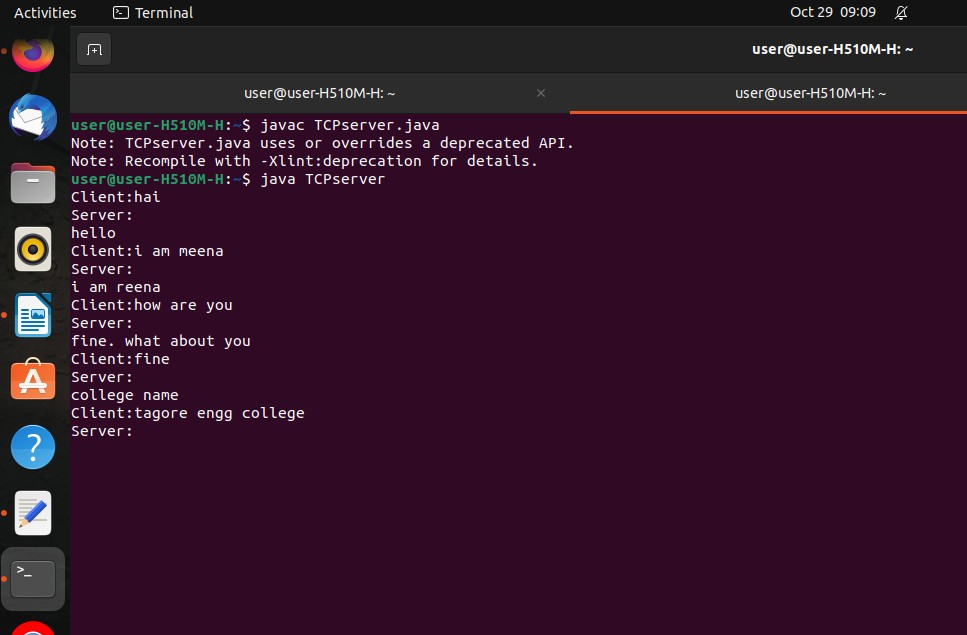
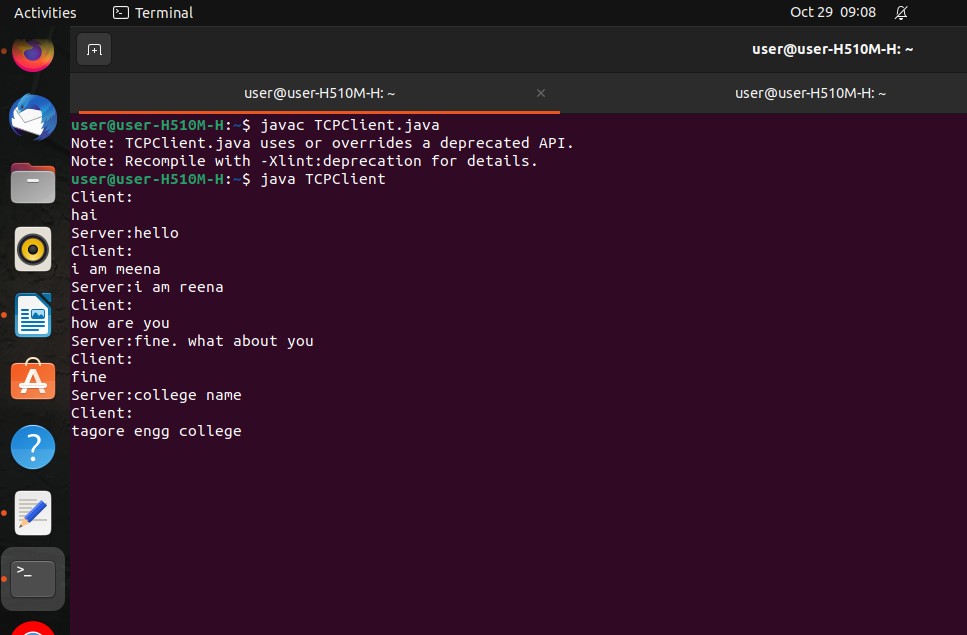
} catch (IOException e) {

System.out.println(e);

}

}

}

**OUTPUT:**

**PROGRAM:**

**dnsserver.java:**

import java.io.\*;

import java.net.\*;

public class dnsserver {

private static int indexOf(String[] array, String str) {

str = str.trim();

for (int i = 0; i < array.length; i++) {

if (array[i].equals(str))

return i;

}

return -1;

}

public static void main(String[] args) throws IOException {

String[] hosts = {"zoho.com", "gmail.com", "google.com", "facebook.com"};

String[] ip = {"172.28.251.59", "172.217.11.5", "172.217.11.14", "31.13.71.36"};

System.out.println("Press Ctrl + C to Quit");

while (true) {

DatagramSocket serversocket = new DatagramSocket(1362);

byte[] senddata = new byte[1021];

byte[] receivedata = new byte[1021];

DatagramPacket recvpack = new DatagramPacket(receivedata, receivedata.length);

serversocket.receive(recvpack);

String sen = new String(recvpack.getData());

InetAddress ipaddress = recvpack.getAddress();

int port = recvpack.getPort();

String capsent;

System.out.println("Request for host " + sen);

if (indexOf(hosts, sen) != -1)

capsent = ip[indexOf(hosts, sen)];

else

capsent = "Host Not Found";

senddata = capsent.getBytes();

DatagramPacket pack = new DatagramPacket(senddata, senddata.length, ipaddress, port);

serversocket.send(pack);

serversocket.close();

}

}

}

**dnsclient.java:**

import java.io.\*;

import java.net.\*;

public class dnsclient {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

DatagramSocket clientsocket = new DatagramSocket();

InetAddress ipaddress;

if (args.length == 0)

ipaddress = InetAddress.getLocalHost();

else

ipaddress = InetAddress.getByName(args[0]);

byte[] senddata = new byte[1024];

byte[] receivedata = new byte[1024];

int portaddr = 1362;

System.out.print("Enter the hostname : ");

String sentence = br.readLine();

senddata = sentence.getBytes();

DatagramPacket pack = new DatagramPacket(senddata, senddata.length, ipaddress, portaddr);

clientsocket.send(pack);

DatagramPacket recvpack = new DatagramPacket(receivedata, receivedata.length);

clientsocket.receive(recvpack);

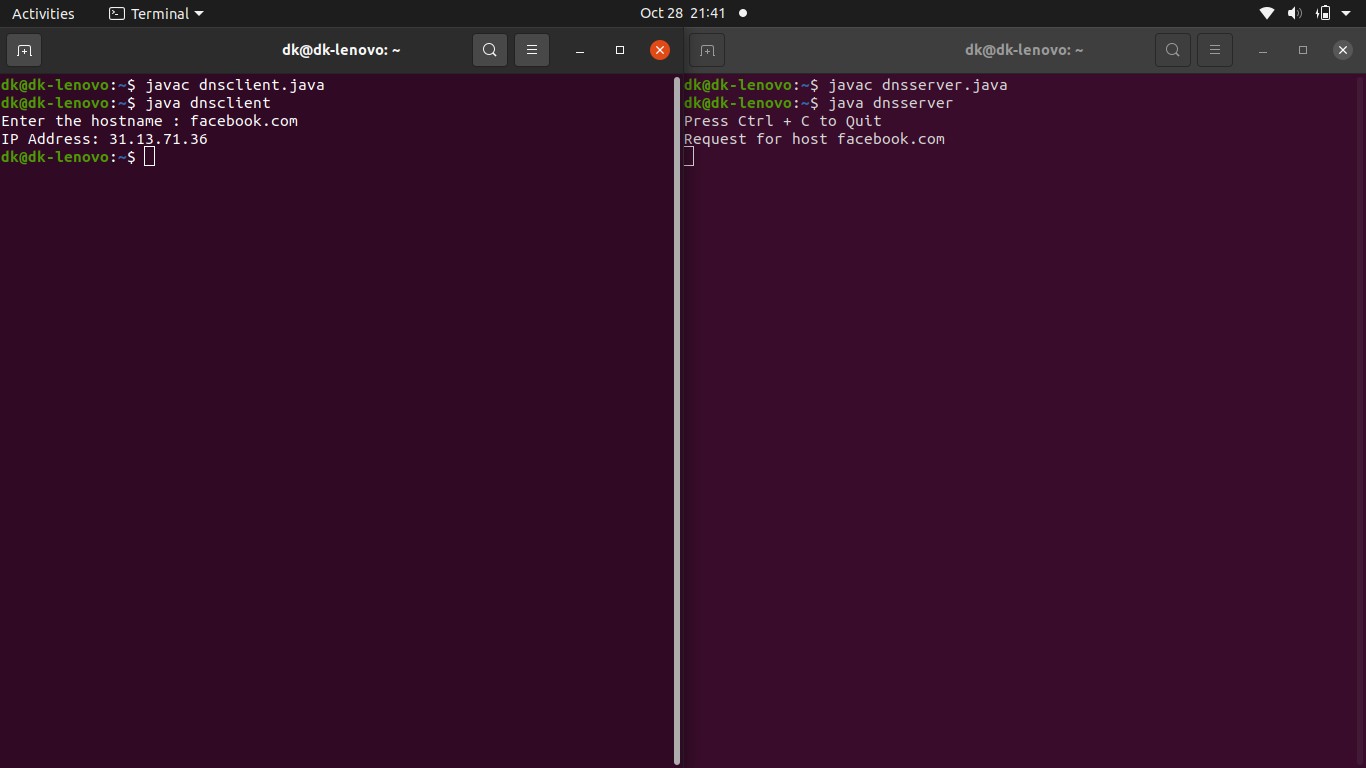
String modified = new String(recvpack.getData());

System.out.println("IP Address: " + modified);

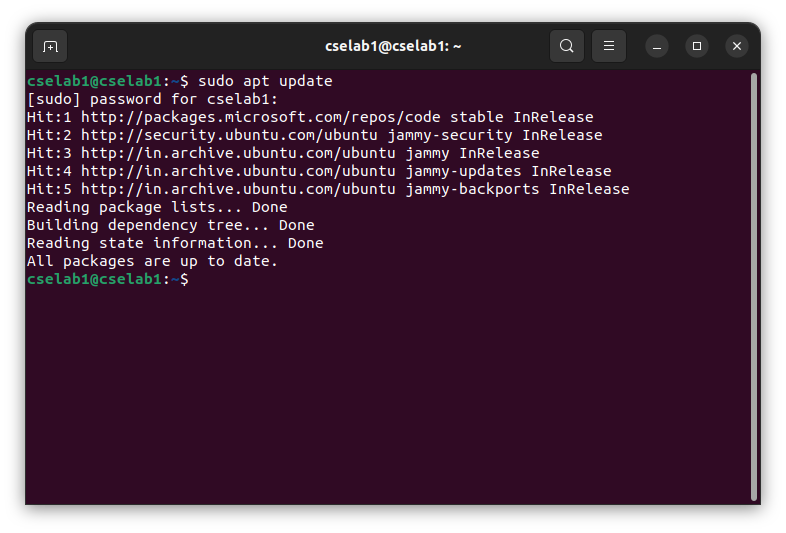
clientsocket.close();

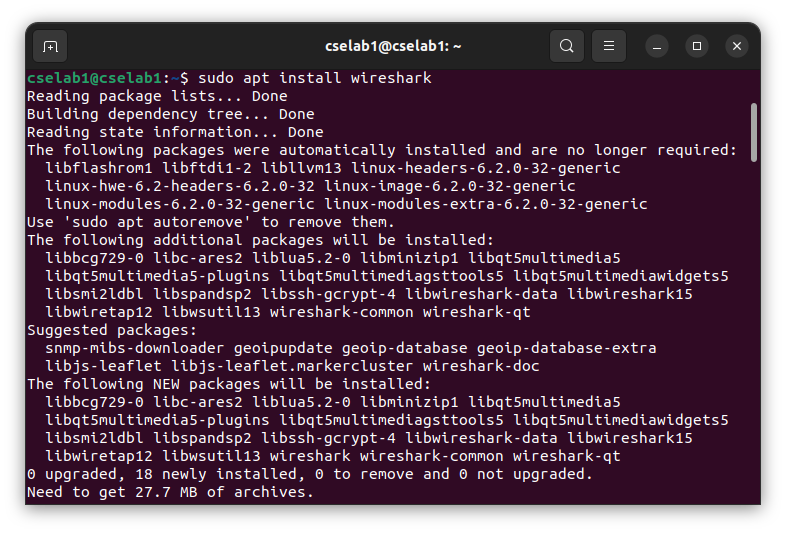
}

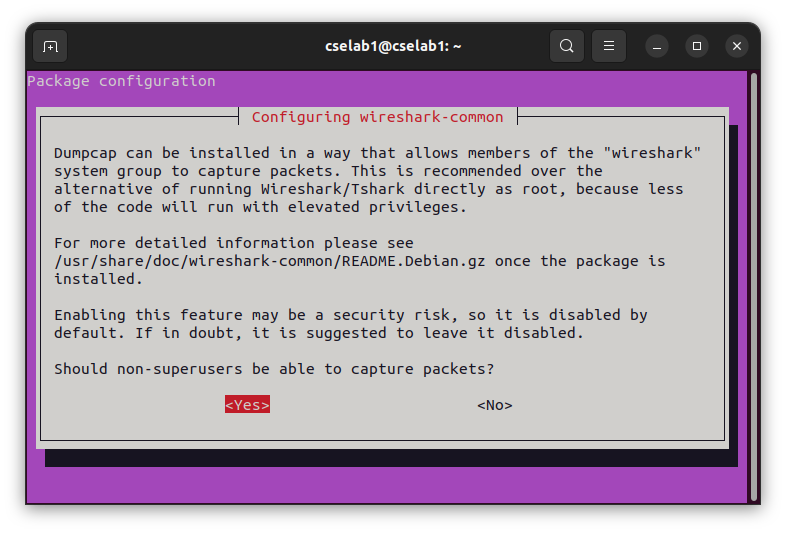
}

**OUTPUT:**

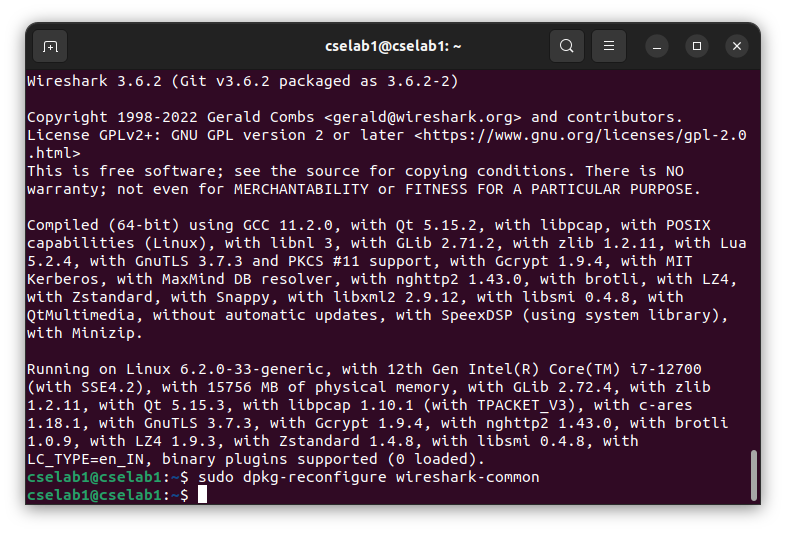
**OUTPUT:**

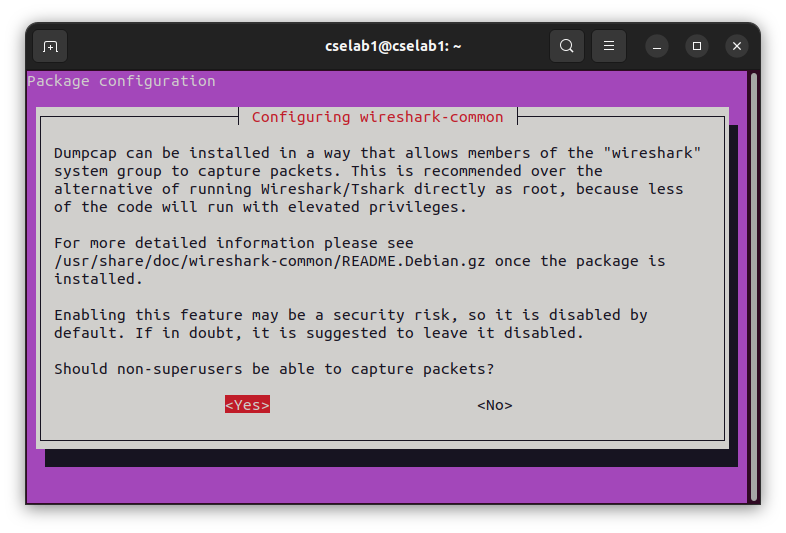


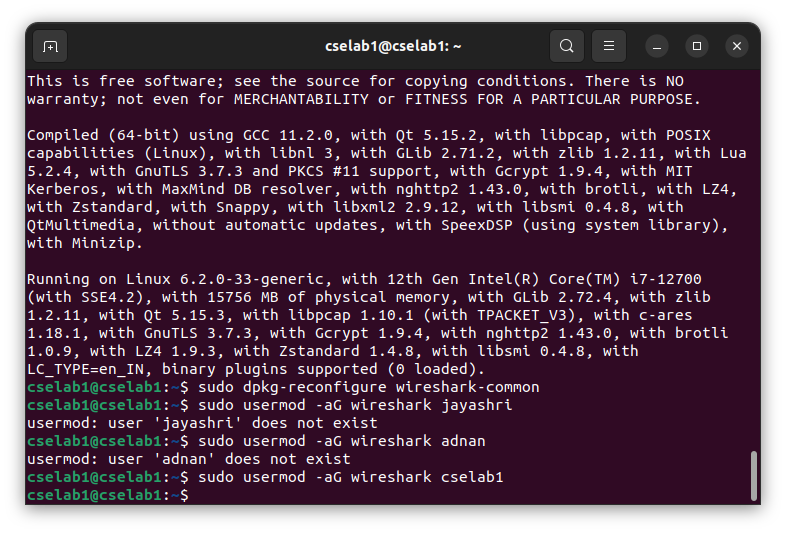


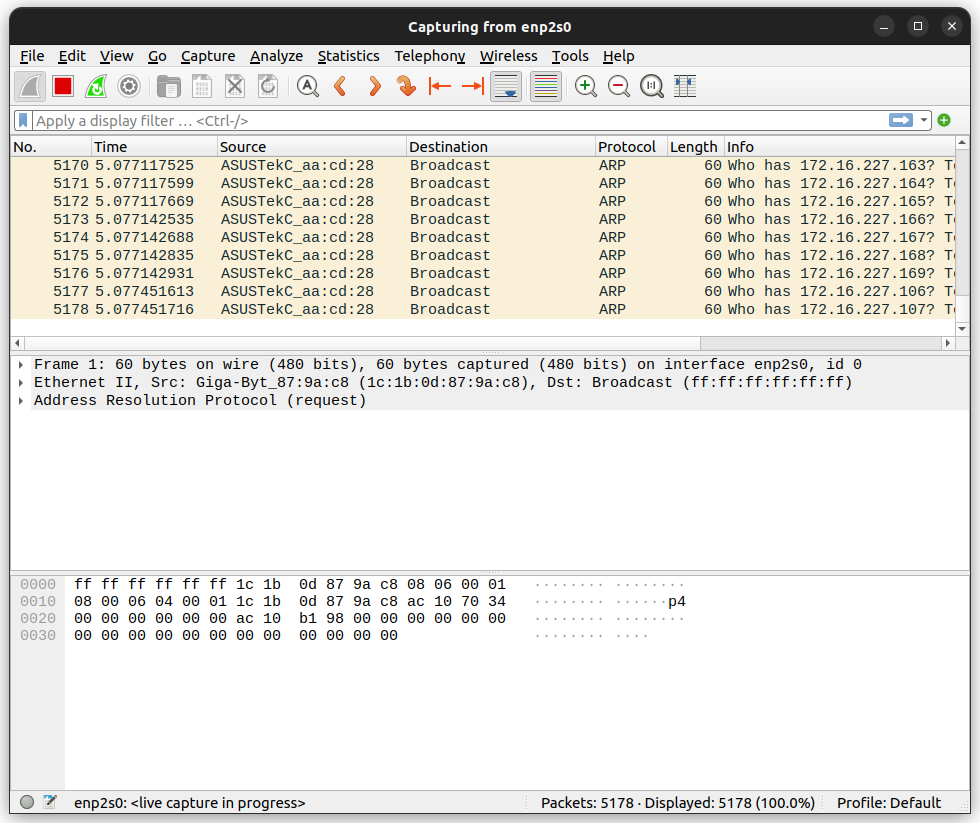


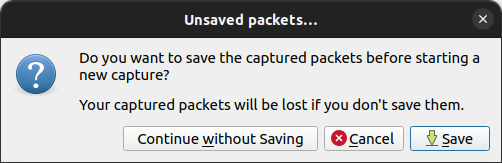


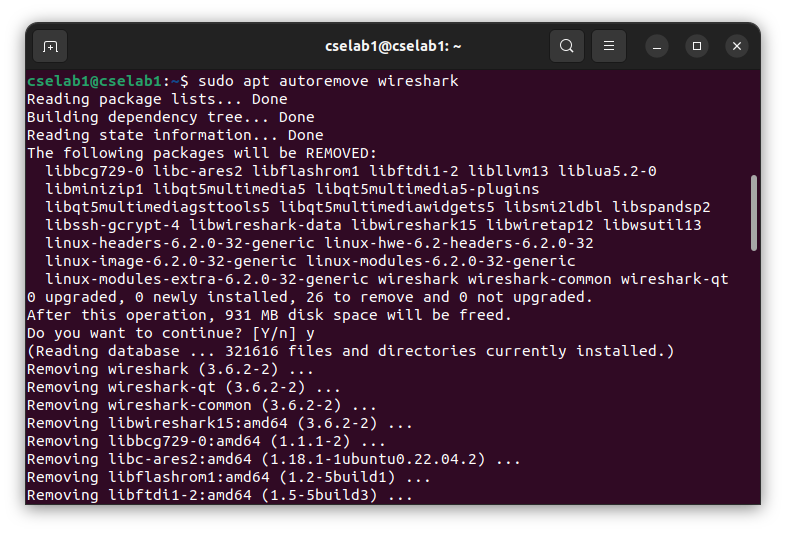








****

****

**PROGRAM:**

**Clientarp.java:**

import java.io.\*;

import java.net.\*;

public class Clientarp {

public static void main(String args[]) {

try {

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

Socket clsct = new Socket("127.0.0.1", 9999);

DataInputStream din = new DataInputStream(clsct.getInputStream());

DataOutputStream dout = new DataOutputStream(clsct.getOutputStream());

System.out.println("Enter the Logical address(IP):");

String str1 = in.readLine();

dout.writeBytes(str1 + '\n');

String str = din.readLine();

System.out.println("The Physical Address is: " + str);

clsct.close();

} catch (Exception e) {

System.out.println(e);

}

}

}

**Serverarp.java:**

import java.io.\*;

import java.net.\*;

public class Serverarp {

public static void main(String args[]) {

try {

ServerSocket obj = new ServerSocket(9999);

Socket obj1 = obj.accept();

while (true) {

DataInputStream din = new DataInputStream(obj1.getInputStream());

DataOutputStream dout = new DataOutputStream(obj1.getOutputStream());

String str = din.readLine();

String ip[] = { "165.165.80.80", "165.165.79.1" };

String mac[] = { "6A:08:AA:C2", "8A:BC:E3:FA" };

boolean found = false;

for (int i = 0; i < ip.length; i++) {

if (str.equals(ip[i])) {

dout.writeBytes(mac[i] + '\n');

found = true;

break;

}

}

if (!found) {

dout.writeBytes("MAC not found" + '\n');

}

}

} catch (Exception e) {

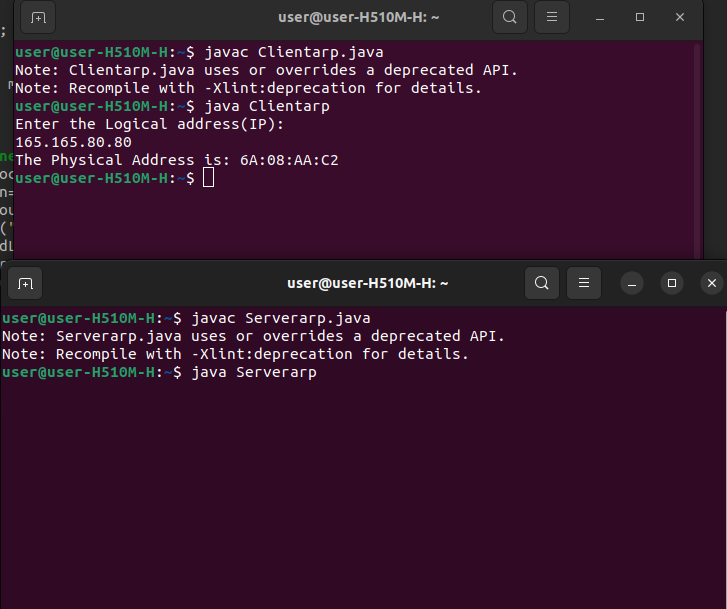
System.out.println(e);

}

}

}

**OUTPUT:**



**PROGRAM:**

**Clientrarp.java:**

import java.io.\*;

import java.net.\*;

public class Clientrarp {

public static void main(String args[]) {

try {

DatagramSocket client = new DatagramSocket();

InetAddress addr = InetAddress.getByName("127.0.0.1");

byte[] sendbyte = new byte[1024];

byte[] receivebyte = new byte[1024];

BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the Physical address (MAC):");

String str = in.readLine();

sendbyte = str.getBytes();

DatagramPacket sender = new DatagramPacket(sendbyte, sendbyte.length, addr, 1309);

client.send(sender);

DatagramPacket receiver = new DatagramPacket(receivebyte, receivebyte.length);

client.receive(receiver);

String s = new String(receiver.getData());

System.out.println("The Logical Address is(IP): " + s.trim());

client.close();

} catch (Exception e) {

System.out.println(e);

}

}

}

**Serverrarp.java:**

import java.io.\*;

import java.net.\*;

public class Serverrarp {

public static void main(String args[]) {

try {

DatagramSocket server = new DatagramSocket(1309);

while (true) {

byte[] sendbyte = new byte[1024];

byte[] receivebyte = new byte[1024];

DatagramPacket receiver = new DatagramPacket(receivebyte, receivebyte.length);

server.receive(receiver);

String str = new String(receiver.getData());

String s = str.trim();

InetAddress addr = receiver.getAddress();

int port = receiver.getPort();

String ip[] = { "165.165.80.80", "165.165.79.1" };

String mac[] = { "6A:08:AA:C2", "8A:BC:E3:FA" };

boolean found = false;

for (int i = 0; i < ip.length; i++) {

if (s.equals(mac[i])) {

sendbyte = ip[i].getBytes();

DatagramPacket sender = new DatagramPacket(sendbyte, sendbyte.length, addr, port);

server.send(sender);

found = true;

break;

}

}

if (!found) {

System.out.println("MAC not found");

}

}

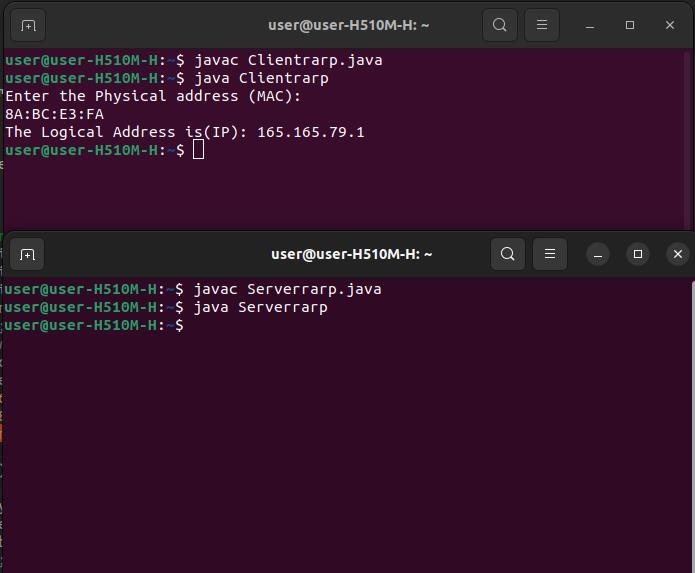
} catch (Exception e) {

System.out.println(e);

}

}

}

**OUTPUT:**

**DATE:**

**EXP.NO.:**07

**STUDY OF NETWORK SIMULATOR (NS) AND SIMULATION OF CONGESTION CONTROL ALGORITHMS USING NS**

**AIM:**

To Study Network Simulator (NS) and Simulate Congestion Control Algorithms using NS.

**PRE LAB DISCUSSION:**

**NETWORK SIMULATOR (NS2):**

**NS Overview:**

* NS Status
* Periodical release (ns-2.26, Feb 2003)
* Platform support
* FreeBSD, Linux, Solaris, Windows, and Mac

**NS Functionalities:**

Routing, Transportation, Traffic sources, Queuing disciplines, QoS

**Congestion Control Algorithms:**

* Slow start
* Additive increase/multiplicative decrease
* Fast retransmit and Fast recovery

**Case Study:** A simple Wireless network.

Ad hoc routing, mobile IP, sensor-MAC Tracing, visualization, and various utilities NS (Network Simulators)

Most commercial simulators are GUI-driven, while some network simulators are CLI-driven. The network model/configuration describes the state of the network (nodes, routers, switches, links) and the events (data transmissions, packet errors, etc.). An important output of simulations is the trace files. Trace files log every packet, every event that occurred in the simulation and are used for analysis. Network simulators can also provide other tools to facilitate visual analysis of trends and potential trouble spots.

Most network simulators use discrete event simulation, in which a list of pending "events" is stored, and those events are processed in order, with some events triggering future events—such as the event of the arrival of a packet at one node triggering the event of the arrival of that packet at a downstream node.

Simulation of networks is a very complex task. For example, if congestion is high, then estimation of the average occupancy is challenging because of high variance. To estimate the likelihood of a buffer overflow in a network, the time required for an accurate answer can be extremely large. Specialized techniques such as "control variates" and "importance sampling" have been developed to speed simulation.

**Examples of network simulators:**

There are many both free/open-source and proprietary network simulators. Examples of notable network simulation software are, ordered after how often they are mentioned in research papers:

1. ns (open source)

2. OPNET (proprietary software)

3. NetSim (proprietary software)

**Uses of network simulators:**

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers, and data links, network simulators are relatively fast and inexpensive. They allow engineers, researchers to test scenarios that might be particularly difficult or expensive to emulate using real hardware - for instance, simulating a scenario with several nodes or experimenting with a new protocol in the network. Network simulators are particularly useful in allowing researchers to test new networking protocols or changes to existing protocols in a controlled and reproducible environment. A typical network simulator encompasses a wide range of networking technologies and can help users build complex networks from basic building blocks such as a variety of nodes and links. With the help of simulators, one can design hierarchical networks using various types of nodes like computers, hubs, bridges, routers, switches, links, mobile units, etc.

Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP, etc., and Local Area Network (LAN) technologies like Ethernet, token rings, etc., can all be simulated with a typical simulator, and the user can test, analyze various standard results apart from devising some novel protocol or strategy for routing, etc. Network simulators are also widely used to simulate battlefield networks in Network-centric warfare.

There are a wide variety of network simulators, ranging from the very simple to the very complex. Minimally, a network simulator must enable a user to represent a network topology, specifying the nodes on the network, the links between those nodes, and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to handle traffic in a network. Graphical applications allow users to easily visualize the workings of their simulated environment. Text-based applications may provide a less intuitive interface but may permit more advanced forms of customization.

**Packet Loss:**

Packet loss occurs when one or more packets of data traveling across a computer network fail to reach their destination. Packet loss is distinguished as one of the three main error types encountered in digital communications; the other two being bit error and spurious packets caused due to noise.

Packets can be lost in a network because they may be dropped when a queue in the network node overflows. The amount of packet loss during the steady state is another important property of a congestion control scheme. The larger the value of packet loss, the more difficult it is for transport layer protocols to maintain high bandwidths. The sensitivity to the loss of individual packets, as well as to the frequency and patterns of loss among longer packet sequences, is strongly dependent on the application itself.

**Throughput:**

Throughput is the main performance measure characteristic and is the most widely used. In communication networks, such as Ethernet or packet radio, throughput or network throughput is the average rate of successful message delivery over a communication channel. Throughput is usually measured in bits per second (bit/s or bps) and sometimes in data packets per second or data packets per time slot. These measures determine how soon the receiver can get a certain amount of data sent by the sender. It is determined as the ratio of the total data received to the end-to-end delay. Throughput is an important factor that directly impacts network performance.

**Delay:**

Delay is the time elapsed while a packet travels from one point, e.g., source premise or network ingress to destination premise or network degrees. The larger the value of delay, the more difficult it is for transport layer protocols to maintain high bandwidths. We will calculate end-to-end delay.

**Queue Length:**

A queuing system in networks can be described as packets arriving for service, waiting for service if it is not immediate, and if having waited for service, leaving the system after being served. Thus queue length is a very important characteristic to determine how well the active queue management of the congestion control algorithm has been working.

**Congestion Control Algorithms:**

Slow-start is used in conjunction with other algorithms to avoid sending more data than the network is capable of transmitting, that is, to avoid causing network congestion. The additive increase/multiplicative decrease (AIMD) algorithm is a feedback control algorithm. AIMD combines linear growth of the congestion window with an exponential reduction when congestion takes place. Multiple flows using AIMD congestion control will eventually converge to use equal amounts of a contended link. Fast Retransmit is an enhancement to TCP that reduces the time a sender waits before retransmitting a lost segment.

**PROGRAM:**

#include <wifi\_lte/wifi\_lte\_rtable.h>

struct r\_hist\_entry \*elm, \*elm2;

int num\_later = 1;

elm = STAILQ\_FIRST(&r\_hist\_);

while (elm != NULL && num\_later <= num\_dup\_acks\_) {

num\_later;

elm = STAILQ\_NEXT(elm, linfo\_);

}

if (elm != NULL) {

elm = findDataPacketInRecvHistory(STAILQ\_NEXT(elm, linfo\_));

if (elm != NULL) {

elm2 = STAILQ\_NEXT(elm, linfo\_);

while (elm2 != NULL) {

if (elm2->seq\_num\_ < seq\_num && elm2->t\_recv\_ < time) {

STAILQ\_REMOVE(&r\_hist\_, elm2, r\_hist\_entry, linfo\_);

delete elm2;

} else {

elm = elm2;

}

elm2 = STAILQ\_NEXT(elm, linfo\_);

}

}

}

void DCCPTFRCAgent::removeAcksRecvHistory() {

struct r\_hist\_entry \*elm1 = STAILQ\_FIRST(&r\_hist\_);

struct r\_hist\_entry \*elm2;

int num\_later = 1;

while (elm1 != NULL && num\_later <= num\_dup\_acks\_) {

num\_later;

elm1 = STAILQ\_NEXT(elm1, linfo\_);

}

if (elm1 == NULL) return;

elm2 = STAILQ\_NEXT(elm1, linfo\_);

while (elm2 != NULL) {

if (elm2->type\_ == DCCP\_ACK) {

STAILQ\_REMOVE(&r\_hist\_, elm2, r\_hist\_entry, linfo\_);

delete elm2;

} else {

elm1 = elm2;

}

elm2 = STAILQ\_NEXT(elm1, linfo\_);

}

}

inline r\_hist\_entry \*DCCPTFRCAgent::findDataPacketInRecvHistory(r\_hist\_entry \*start) {

while (start != NULL && start->type\_ == DCCP\_ACK)

start = STAILQ\_NEXT(start, linfo\_);

return start;

}

**RESULT:**

Thus, we have studied the Network Simulator (NS) and simulated Congestion Control Algorithms using NS.

**PROGRAM:**

set ns [new Simulator]

set nr [open thro\_dt.tr w]

$ns trace-all $nr

set nf [open thro.nam w]

$ns namtrace-all $nf

proc finish { } {

global ns nr nf

$ns flush-trace

close $nf

close $nr

exec nam thro.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

set n6 [$ns node]

set n7 [$ns node]

$ns duplex-link $n0 $n3 1Mb 10ms DropTail

$ns duplex-link $n1 $n3 1Mb 10ms DropTail

$ns duplex-link $n2 $n3 1Mb 10ms DropTail

$ns duplex-link $n3 $n4 1Mb 10ms DropTail

$ns duplex-link $n4 $n5 1Mb 10ms DropTail

$ns duplex-link $n4 $n6 1Mb 10ms DropTail

$ns duplex-link $n4 $n7 1Mb 10ms DropTail

$ns duplex-link-op $n0 $n3 orient right-up

$ns duplex-link-op $n1 $n3 orient right

$ns duplex-link-op $n2 $n3 orient right-down

$ns duplex-link-op $n3 $n4 orient middle

$ns duplex-link-op $n4 $n5 orient right-up

$ns duplex-link-op $n4 $n7 orient right-down

$ns duplex-link-op $n6 $n4 orient left

set udp0 [new Agent/UDP]

$ns attach-agent $n2 $udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0

set null0 [new Agent/Null]

$ns attach-agent $n5 $null0

$ns connect $udp0 $null0

set udp1 [new Agent/UDP]

$ns attach-agent $n1 $udp1

set cbr1 [new Application/Traffic/CBR]

$cbr1 set packetSize\_ 500

$cbr1 set interval\_ 0.005

$cbr1 attach-agent $udp1

set null0 [new Agent/Null]

$ns attach-agent $n6 $null0

$ns connect $udp1 $null0

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set cbr2 [new Application/Traffic/CBR]

$cbr2 set packetSize\_ 500

$cbr2 set interval\_ 0.005

$cbr2 attach-agent $tcp0

set tcpsink0 [new Agent/TCPSink]

$ns attach-agent $n7 $tcpsink0

$ns connect $tcp0 $tcpsink0

$udp0 set fid\_ 1

$udp1 set fid\_ 2

$tcp0 set fid\_ 3

$ns color 1 Red

$ns color 2 Green

$ns color 3 Blue

$ns at 0.2 "$cbr0 start"

$ns at 3.5 "$cbr0 stop"

$ns at 0.3 "$cbr1 start"

$ns at 4.5 "$cbr1 stop"

$ns at 0.4 "$cbr2 start"

$ns at 4.5 "$cbr2 stop"

$ns at 5.0 "finish"

$ns run

Create a simulation obect

2. Set routing protocol to routing

3. Trace packets and all links onto NAM trace and to trace file

4. Create right nodes

5. Describe their layout topology

6. Add a sink agent to node

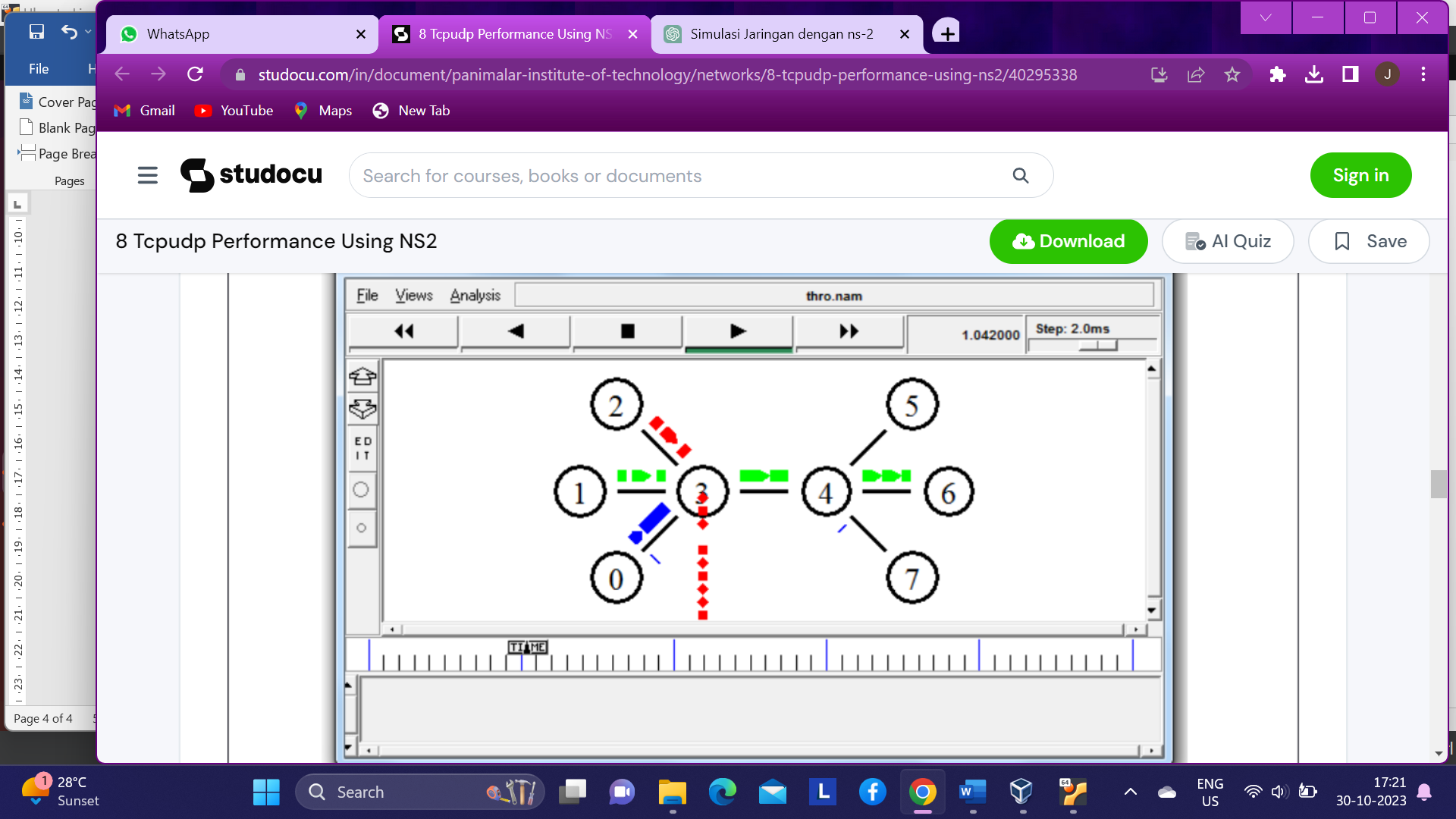
7. Connect source and sink.

8. Observe the traffic route when link is up and down

9. View the simulated events and trace file analyze it

10. Start the scheduler

**OUTPUT:**

****

**PROGRAM:**

set ns [new Simulator]

set nr [open thro.tr w]

$ns trace-all $nr

set nf [open thro.nam w]

$ns namtrace-all $nf

proc finish { } {

global ns nr nf

$ns flush-trace

close $nf

close $nr

exec nam thro.nam &

exit 0

}

for { set i 0 } { $i < 12} { incr i 1 } {

set n($i) [$ns node]}

for {set i 0} {$i < 8} {incr i} {

$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }

$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail

$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail

$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail

$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail

$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail

$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail

set udp0 [new Agent/UDP]

$ns attach-agent $n(0) $udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0

set null0 [new Agent/Null]

$ns attach-agent $n(5) $null0

$ns connect $udp0 $null0

set udp1 [new Agent/UDP]

$ns attach-agent $n(1) $udp1

set cbr1 [new Application/Traffic/CBR]

$cbr1 set packetSize\_ 500

$cbr1 set interval\_ 0.005

$cbr1 attach-agent $udp1

set null0 [new Agent/Null]

$ns attach-agent $n(5) $null0

$ns connect $udp1 $null0

$ns rtproto DV

$ns rtmodel-at 10.0 down $n(11) $n(5)

$ns rtmodel-at 15.0 down $n(7) $n(6)

$ns rtmodel-at 30.0 up $n(11) $n(5)

$ns rtmodel-at 20.0 up $n(7) $n(6)

$udp0 set fid\_ 1

$udp1 set fid\_ 2

$ns color 1 Red

$ns color 2 Green

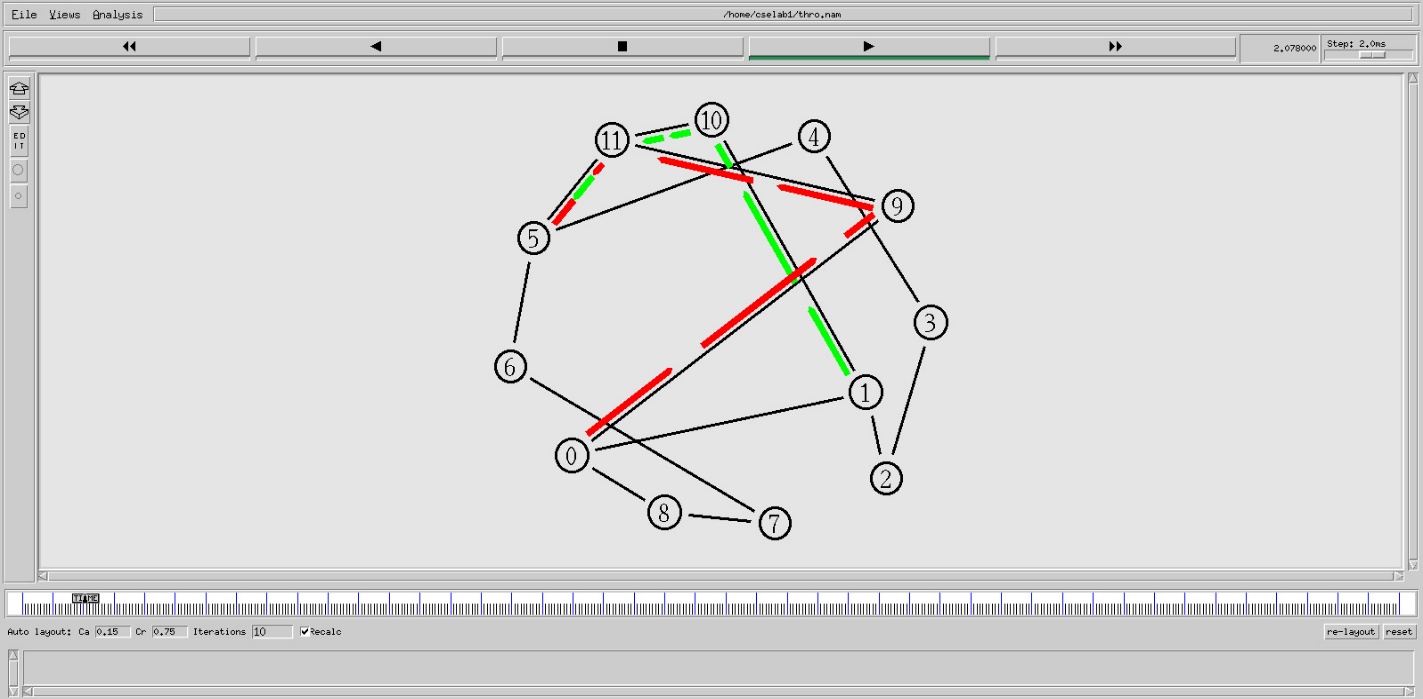
$ns at 1.0 "$cbr0 start"

$ns at 2.0 "$cbr1 start"

$ns at 45 "finish"

$ns run

**OUTPUT:**

****

**PROGRAM:**

set ns [new Simulator]

set nr [open thro.tr w]

$ns trace-all $nr

set nf [open thro.nam w]

$ns namtrace-all $nf

proc finish { } {

global ns nr nf

$ns flush-trace

close $nf

close $nr

exec nam thro.nam &

exit 0

}

for { set i 0 } { $i < 12} { incr i 1 } {

set n($i) [$ns node]}

for {set i 0} {$i < 8} {incr i} {

$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }

$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail

$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail

$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail

$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail

$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail

$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail

set udp0 [new Agent/UDP]

$ns attach-agent $n(0) $udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0

set null0 [new Agent/Null]

$ns attach-agent $n(5) $null0

$ns connect $udp0 $null0

set udp1 [new Agent/UDP]

$ns attach-agent $n(1) $udp1

set cbr1 [new Application/Traffic/CBR]

$cbr1 set packetSize\_ 500

$cbr1 set interval\_ 0.005

$cbr1 attach-agent $udp1

set null0 [new Agent/Null]

$ns attach-agent $n(5) $null0

$ns connect $udp1 $null0

$ns rtproto LS

$ns rtmodel-at 10.0 down $n(11) $n(5)

$ns rtmodel-at 15.0 down $n(7) $n(6)

$ns rtmodel-at 30.0 up $n(11) $n(5)

$ns rtmodel-at 20.0 up $n(7) $n(6)

$udp0 set fid\_ 1

$udp1 set fid\_ 2

$ns color 1 Red

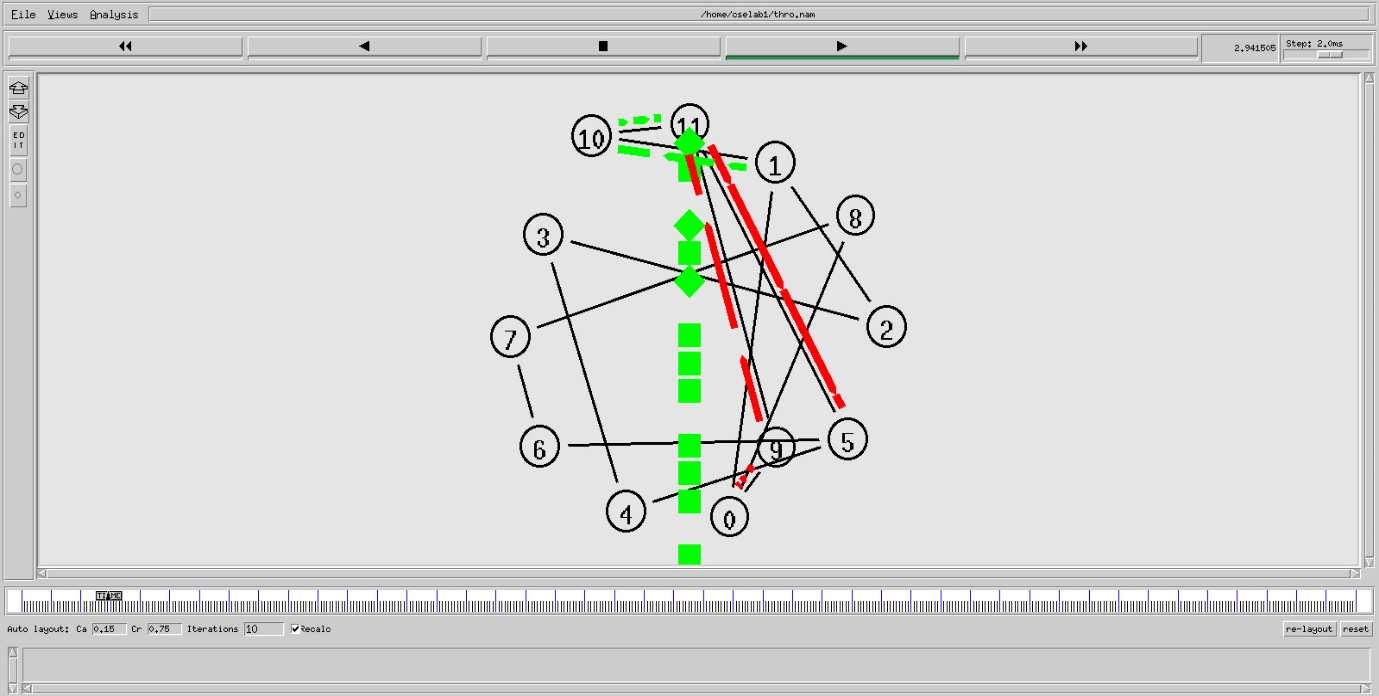
$ns color 2 Green

$ns at 1.0 "$cbr0 start"

$ns at 2.0 "$cbr1 start"

$ns at 45 "finish"

$ns run

**OUTPUT:**

**PROGRAM:**

CRC.java:

import java.io.\*;  
class CRC  
{  
public static void main(String args[]) throws IOException  
{  
BufferedReader br = new BufferedReader(new  
InputStreamReader(System.in));  
System.out.println("Enter Generator:");  
String gen = br.readLine();  
System.out.println("Enter Data:");  
String data = br.readLine();  
String code = data;  
while(code.length() < (data.length() + gen.length() - 1))  
code = code + "0";  
code = data + div(code,gen);  
System.out.println("The transmitted Code Word is: " + code);  
System.out.println("Please enter the received Code Word: ");  
String rec = br.readLine();  
if(Integer.parseInt(div(rec,gen)) == 0)  
System.out.println("The received code word contains no errors.");  
else  
System.out.println("The received code word contains errors.");

}   
static String div(String num1,String num2)  
{  
int pointer = num2.length();  
String result = num1.substring(0, pointer);  
String remainder = "";  
for(int i = 0; i < num2.length(); i++)  
{  
if(result.charAt(i) == num2.charAt(i))  
remainder += "0";  
else  
remainder += "1";  
}  
while(pointer < num1.length())  
{   
if(remainder.charAt(0) == '0')  
{  
remainder = remainder.substring(1, remainder.length());  
remainder = remainder + String.valueOf(num1.charAt(pointer));  
pointer++;  
}  
result = remainder;  
remainder = "";  
for(int i = 0; i < num2.length(); i++)  
{  
if(result.charAt(i) == num2.charAt(i))  
remainder += "0";  
else  
remainder += "1";  
}  
}  
return remainder.substring(1,remainder.length());  
}  
}

**OUTPUT:**

