**MADRAS INSTITUTE OF TECHNOLOGY ANNA UNIVERSITY**

**DEPARTMENT OF INFORMATION TECHNOLOGY IT5511 – COMPUTER NETWORKS LABORATORY**

**RECORD**

**REGISTER NUMBER : 2020506081**

**NAME : SANJAY T**

**SEMESTER : 5**

# DEPARTMENT OF INFORMATION TECHNOLOGY ANNA UNIVERSITY, MIT CAMPUS

CHROMEPET, CHENNAI – 600 044

# BONAFIDE CERTIFICATE

Certified that the bonafide record of the practical work done by

Mr/~~Ms~~. Sanjay T , Register Number: 2020506081 of **Fifth** Semester **B.Tech Information Technology** during the academic period from **August 2022 to December 2022** in the IT5511 – Computer Networks Laboratory.

Date: Course Instructor: Shanmuga priya .R

Examiner:

# TABLE OF CONTENTS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No** | **DATE** | **TITLE** | **PAGE NO** | **SIGN** |
| 1. |  |  |  |  |
| 2. |  |  |  |  |
| 3. |  |  |  |  |
| 4. |  |  |  |  |
| 5. |  |  |  |  |
| 6. |  |  |  |  |
| 7. |  |  |  |  |
| 8. |  |  |  |  |
| 9. |  |  |  |  |
| 10. |  |  |  |  |
| 11. |  |  |  |  |
| 12. |  |  |  |  |
| 13. |  |  |  |  |
| 14. |  |  |  |  |
| 15. |  |  |  |  |
| 16. |  |  |  |  |
| 17. |  |  |  |  |

|  |  |
| --- | --- |
| **Ex. No: 1** | **STUDY OF BASIC NETWORK COMMANDS** |
| **Date: 29/08/2022** |

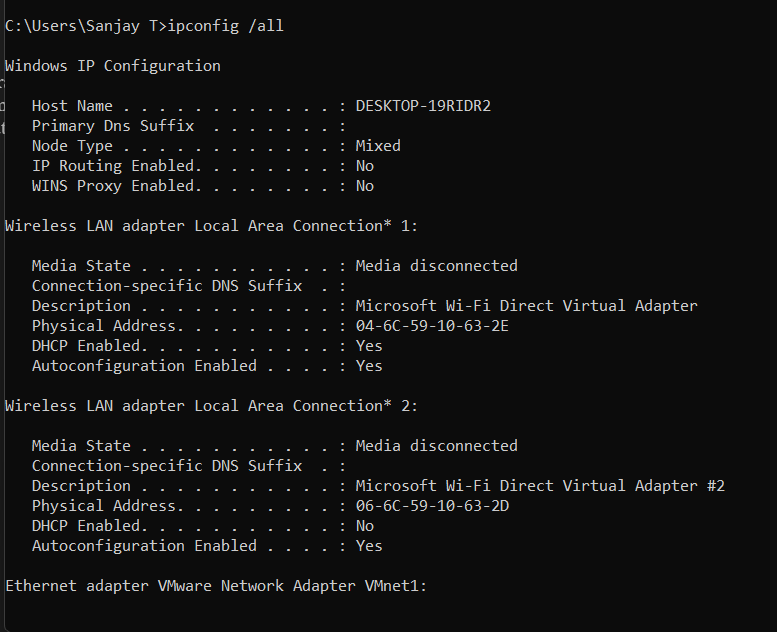
**Aim:**

To get introduced on basic commands related to Networking on Windows.

# Commands and output:

# Ipconfig:

Displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings. Used without parameters, **ipconfig** displays the IP address, subnet mask, and default gateway for all adapters.



**2) Ping command:**

The **P**ing command is used to check the destination IP address to be reached and record the results. The ping command displays whether the destination responded and how long it took to receive a reply. If there is an error in the delivery to the destination, the ping command displays an error message.

Text

Description automatically generated

**3) Traceroute command:**

Traceroute command (tracert) is a utility designed for displaying the time it takes for a packet of information to travel between a local computer and a destination IP address or domain. After running a traceroute command, the results displayed are a list of the 'hops' that data packets take along their path to the designated IP address or domain.

Graphical user interface, text

Description automatically generated

**4 ) Pathping command:**

The pathping command is a route tracing tool that combines features of the ping and tracert commands with additional information that neither of those tools provides. The pathping command sends packets to each router on the way to a final destination over a period of time, and then computes results based on the packets returned from each hop. Since the command shows the degree of packet loss at any given router or link, it is easy to determine which routers or links might be causing network problems.

Text

Description automatically generated

**5) Arp command:**

The address resolution protocol (arp) is a protocol used by the [Internet Protocol (IP)](http://www.erg.abdn.ac.uk/~gorry/eg3561/inet-pages/ip.html) , specifically IPv4, to map [IP network addresses](http://www.erg.abdn.ac.uk/~gorry/eg3561/inet-pages/ip-address.html) to the hardware addresses used by a data link protocol. The protocol operates below the network layer as a part of the interface between the OSI network and OSI link layer. It is used when [IPv4 is used over Ethernet.](http://www.erg.abdn.ac.uk/~gorry/eg3561/inet-pages/ip-enet.html)

The term address resolution refers to the process of finding an address of a computer in a network. The address is "resolved" using a protocol in which a piece of information is sent by a client process executing on the local computer to a server process executing on a remote computer. The information received by the server allows the server to uniquely identify the network system for which the address was required and therefore to provide the required address. The address resolution procedure is completed when the client receives a response from the server containing the required address.

Text

Description automatically generated with medium confidence

**6) Hostname command:**

Display the hostname of the machine the command is being run on**.**

Text

Description automatically generated

**7) Netstat command:**

The netstat command is used to display the [TCP/IP](http://www.computerhope.com/jargon/t/tcpip.htm) network protocol statistics and information.

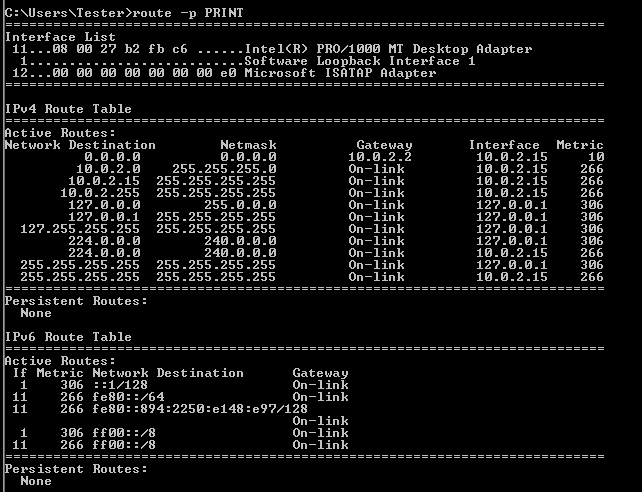
**A picture containing graphical user interface

Description automatically generatedText

Description automatically generated**

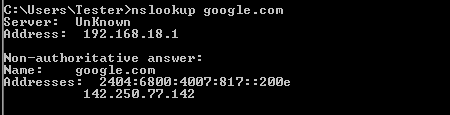
**8) Route command:**

Command to manually configure the routes in the routing table.



**9) Nslookup command:**

MS-DOS utility that enables a user to look up an IP address of a domain or host on a network.



**10) Nbstat command:**

Displays NetBIOS over TCP/IP (NetBT) protocol statistics, NetBIOS name tables for both the local computer and remote computers, and the NetBIOS name cache. Nbtstat allows a refresh of the NetBIOS name cache and the names registered with Windows Internet Name Service (WINS).

Text

Description automatically generated

**11) Netsh:**

Netsh command is powerful utility to view and configure almost all of network adapters. When passed without arguments it lands inside a interactive shell.

Text

Description automatically generated

**12) Getmac command:**

The getmac command provides an easy way to find the MAC address of your device. Prints more than one MAC address device has multiple network adapters.

**Text

Description automatically generated**

**13) Net command:**

The net command allows user to manage many different aspects of a network and its settings such as network shares, users and print jobsetc..

Text

Description automatically generated

**14) Systeminfo command:**

Systeminfo command collects all the important details of a system including processor details, model of the system and network interfaces and puts them in a readable format.

Text

Description automatically generated

**15) Ipconfig /flushdns:**

This command is only needed if you’re having trouble with your networks DNS configuration.

Text

Description automatically generated with medium confidence

**Result:**

Hence, some basic windows networking commands were tested successfully.

|  |  |
| --- | --- |
| **Ex. No: 2** | **SOCKET PROGRAMMING USING TCP – ECHO AND CHAT** |
| **Date: 12/09/2022** |

**Aim:**

To create echo and chat application by implementing sockets using TCP protocol in java programs.

1. **Echo server and client:**

**Algorithm:**

Server side:

Client side:

**Source Code:**

Server side:

import java.io.\*;

import java.net.\*;

public class TCPechoserver {

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

ServerSocket ss = new ServerSocket(8080);

Socket s = ss.accept();

System.out.println("[INFO] : Client connected");

InputStreamReader socIn = new InputStreamReader(s.getInputStream());

BufferedReader in = new BufferedReader(socIn);

String x;

while (true) {

x = in.readLine();

System.out.println("[CLIENT] : " + x);

if (x.equalsIgnoreCase("terminate")){

System.out.println("[INFO] : Client terminated connection");

break;

}

}

ss.close();

}

Client side:

import java.io.\*;

import java.net.\*;

public class TCPechoclient {

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

Socket s = new Socket("localhost", 8080);

System.out.println("[INFO] : Server connection established");

PrintWriter pr = new PrintWriter(s.getOutputStream(), true);

InputStreamReader sysIn = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(sysIn);

String y;

while (true) {

System.out.print("[SYSTEM] > Enter message:");

y = br.readLine();

if (y.equalsIgnoreCase("terminate")) {

System.out.println("[INFO] : Connection terminated");

pr.println("terminate");

break;

} else

pr.println(y);

}

s.close();

}

}

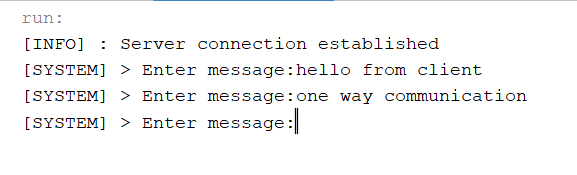
**Output:**

Server:

Text

Description automatically generated

Client:



1. **Chat server and client:**

**Algorithm:**

Server side:

Client side:

**Source Code:**

Server side:

import java.io.\*;

import java.net.\*;

public class TCPchatserver {

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

ServerSocket ss = new ServerSocket(8080);

Socket s = ss.accept();

System.out.println("[INFO] : Client connected");

InputStreamReader socIn = new InputStreamReader(s.getInputStream());

BufferedReader in = new BufferedReader(socIn);

InputStreamReader sysIn = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(sysIn);

PrintWriter pr = new PrintWriter(s.getOutputStream(), true);

String x, y;

System.out.println("[INFO] : Enter 'terminate' to end connection");

while (true) {

x = in.readLine();

if (x.equalsIgnoreCase("terminate")) {

System.out.println("[INFO] : Client terminated connection");

break;

}

System.out.println("[CLIENT] : " + x);

System.out.print("[SYSTEM] > Enter message:");

y = br.readLine();

if (y.equalsIgnoreCase("terminate")) {

System.out.print("[INFO] : Connection terminated");

pr.println("terminate");

break;

} else

pr.println(y);

}

ss.close();

}

}

Client side:

import java.io.\*;

import java.net.\*;

public class TCPchatclient {

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

Socket s = new Socket("localhost", 8080);

System.out.println("[INFO] : Server connection established");

PrintWriter pr = new PrintWriter(s.getOutputStream(), true);

InputStreamReader socIn = new InputStreamReader(s.getInputStream());

BufferedReader in = new BufferedReader(socIn);

InputStreamReader sysIn = new InputStreamReader(System.in);

BufferedReader br = new BufferedReader(sysIn);

String x, y;

System.out.println("[INFO] : Enter 'terminate' to end connection");

while (true) {

System.out.print("[SYSTEM] > Enter message:");

y = br.readLine();

pr.println(y);

if (y.equalsIgnoreCase("terminate")) {

System.out.println("[INFO] : Connection terminated");

break;

}

x = in.readLine();

if (x.equalsIgnoreCase("terminate")){

System.out.println("[INFO] : Server terminated connection");

break;

}

System.out.println("[SERVER] : " + x);

}

s.close();

}

}

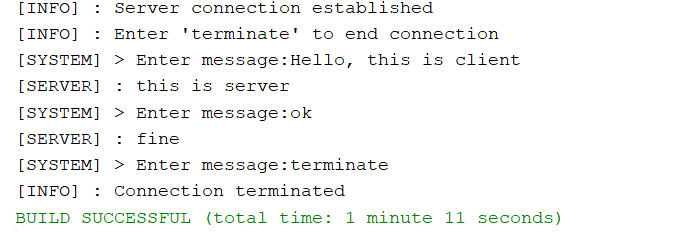
**Output:**

Server:

**Text

Description automatically generated**

Client:



**Result:**

Hence, Echo and Chat application were created and executed successfully with TCP protocol using Java.

|  |  |
| --- | --- |
| **Ex. No: 3** | **SOCKET PROGRAMMING USING UDP – ECHO AND CHAT** |
| **Date: 19/09/2022** |

**Aim:**

To create echo and chat application by implementing sockets using UDP protocol in java programs.

1. **Echo server and client:**

**Algorithm:**

Server side:

Client side:

**Source Code:**

Server side:

import java.io.\*;

import java.net.\*;

public class UDPechoserver {

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

DatagramSocket server = new DatagramSocket(4160);

byte[] buf = new byte[1024];

DatagramPacket packet = new DatagramPacket(buf, buf.length);

System.out.println("[INFO] : Server started");

while (true) {

server.receive(packet);

String str = new String(packet.getData(), 0, packet.getLength());

if (str.equals("terminate")) {

System.out.println("[INFO] : Client terminated connection");

break;

}

System.out.println("[CLIENT] : " + str);

}

server.close();

}

}

Client side:

import java.io.\*;

import java.net.\*;

public class UDPechoclient {

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

DatagramSocket client = new DatagramSocket();

InetAddress add = InetAddress.getByName("localhost");

byte buf[] = new byte[1024];

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

System.out.println("[INFO] : Client started\n[INFO] : Enter 'terminate' to end connection");

while (true) {

System.out.print("[SYSTEM] > Enter message:");

String str = new String(dis.readLine());

buf = str.getBytes();

if (str.equalsIgnoreCase("terminate")) {

System.out.println("[INFO] : Connection terminated");

client.send(new DatagramPacket(buf, str.length(), add, 4160));

break;

}

client.send(new DatagramPacket(buf, str.length(), add, 4160));

}

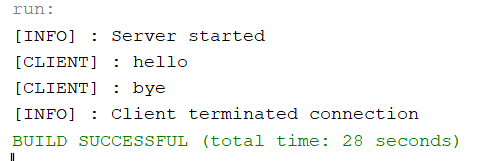
client.close();

}

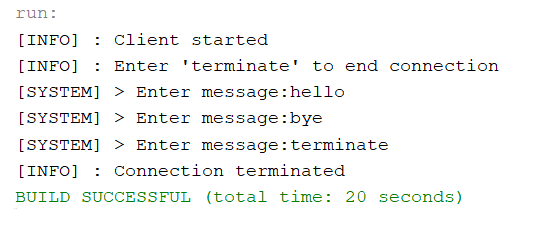
}

**Output :**

Server side:



Client side:



1. **Chat server and client:**

**Algorithm:**

Server side:

Client side:

**Source code:**

Server side:

import java.io.\*;

import java.net.\*;

public class UDPchatserver {

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

int c\_port = 4150, s\_port = 4160;

DatagramSocket server = new DatagramSocket(s\_port);

byte[] buf = new byte[1024];

DatagramPacket packet = new DatagramPacket(buf, buf.length);

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

InetAddress ia = InetAddress.getLocalHost();

System.out.println("Server is Running...");

while (true) {

server.receive(packet);

String str = new String(packet.getData(), 0, packet.getLength());

if (str.equals("STOP")) {

System.out.println("Terminated...");

break;

}

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

String s = new String(br.readLine());

buf = s.getBytes();

server.send(new DatagramPacket(buf, s.length(), ia, c\_port));

}

server.close();

}

} Client side:

import java.io.\*;

import java.net.\*;

public class UDPchatclient {

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

int c\_port = 4150, s\_port = 4160;

DatagramSocket client = new DatagramSocket(c\_port);

InetAddress add = InetAddress.getByName("localhost");

byte buf[] = new byte[1024];

DatagramPacket dp = new DatagramPacket(buf, buf.length);

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

System.out.println("Client is Running... Type 'STOP' to Quit");

while (true) {

String str = new String(dis.readLine());

buf = str.getBytes();

if (str.equals("STOP")) {

System.out.println("Terminated...");

client.send(new DatagramPacket(buf, str.length(), add, s\_port));

break;

}

client.send(new DatagramPacket(buf, str.length(), add, s\_port));

client.receive(dp);

String str2 = new String(dp.getData(), 0, dp.getLength());

System.out.println("Server: " + str2);

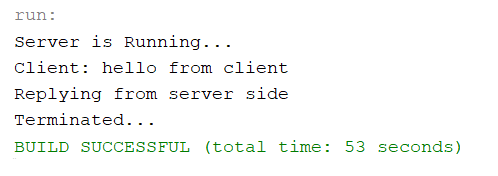
}

client.close();

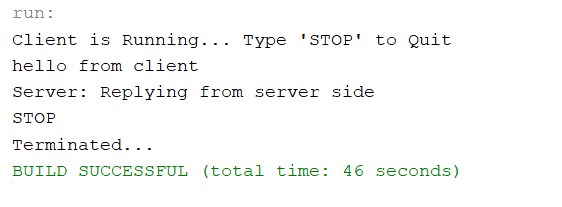
}}

**Output:**

Server side:



Client side:



**Result:**

Hence, Echo and Chat application were created and executed successfully with UDP protocol using Java.

|  |  |
| --- | --- |
| **Ex. No: 4** | **HTTP PROTOCOL – GET AND POST** |
| **Date: 24/09/2022** |

**Aim:**

To create programs to implement HTTP protocol using get and post methods in java.

1. **HTTP GET server and client:**

**Algorithm:**

Server side:

Client side:

**Source Code:**

Server side:

import java.io.\*;

import java.net.\*;

public class HTTPGetServer {

private static final String USER\_AGENT = "Google Chrome";

static String sendGET(String GET\_URL) throws Exception {

URL URLObj = new URL(GET\_URL);

HttpURLConnection con = (HttpURLConnection) URLObj.openConnection();

con.setRequestMethod("GET");

con.setRequestProperty("User-Agent", USER\_AGENT);

int responseCode = con.getResponseCode();

System.out.println("[INFO] : Response Code : " + responseCode);

if (responseCode == HttpURLConnection.HTTP\_OK) {

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

String inputLine;

StringBuffer response = new StringBuffer();

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

response.append(inputLine);

}

in.close();

return(response.toString());

}

else{

System.out.println("[ERROR] : GET request failed");

return (null);

}

}

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

ServerSocket ss=new ServerSocket(6789);

while(true){

Socket conSoc= ss.accept();

BufferedReader ifc =new BufferedReader(new InputStreamReader(conSoc.getInputStream()));

DataOutputStream otc =new DataOutputStream(conSoc.getOutputStream());

String cs=ifc.readLine()+'\n';

System.out.println("[INFO] : Requested URL : "+cs);

String GET\_URL = cs;

otc.writeBytes(sendGET(GET\_URL)+'\n');

System.out.println("[INFO] : GET Request successful");

break;

}

ss.close();

}

}

Client side:

import java.io.\*;

import java.net.\*;

public class HTTPGetClient {

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

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

Socket clientSocket=new Socket("localhost",6789);

DataOutputStream ots=new DataOutputStream(clientSocket.getOutputStream());

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

System.out.print("[SYSTEM] > Enter URL : ");

String sentence = ifu.readLine();

ots.writeBytes(sentence+'\n');

String ms = ifs.readLine();

System.out.println("[SERVER] : GET Resoponse:\n"+ms);

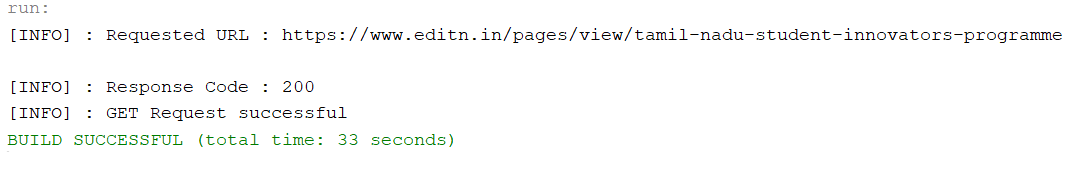
clientSocket.close();

}

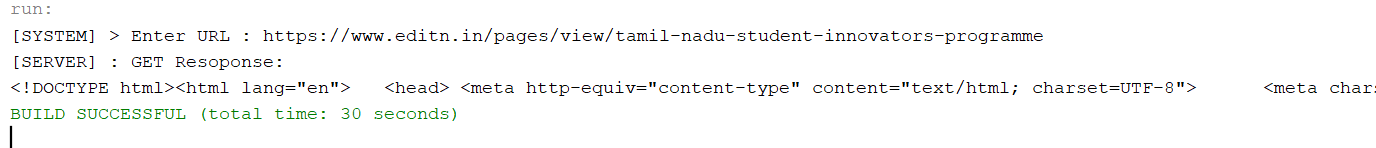
}

**Output:**

Server side:



Client side:



1. **HTTP POST server and client:**

**Algorithm:**

Server side:

Client side:

**Source code:**

Server side:

import java.io.\*;

import java.net.\*;

public class HTTPPostServer {

private static final String USER\_AGENT = "Google Chrome";

static String sendPOST(String POST\_URL) throws IOException{

URL obj = new URL(POST\_URL);

HttpURLConnection con = (HttpURLConnection) obj.openConnection();

con.setRequestMethod("POST");

con.setRequestProperty("User-Agent", USER\_AGENT);

con.setDoOutput(true);

int responseCode = con.getResponseCode();

System.out.println("[INFO] : Response Code : " + responseCode);

if (responseCode == HttpURLConnection.HTTP\_OK) {

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

String inputLine;

StringBuffer response = new StringBuffer();

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

response.append(inputLine);

}

in.close();

return(response.toString());

} else{

System.out.println("[ERROR] : POST Request failed");

return(null);

}

}

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

ServerSocket ss=new ServerSocket(6789);

while(true) {

Socket consoc= ss.accept();

BufferedReader ifc =new BufferedReader(new InputStreamReader(consoc.getInputStream()));

DataOutputStream otc =new DataOutputStream(consoc.getOutputStream());

String ps=ifc.readLine()+'\n';

System.out.println("[INFO] : Requested URL : "+ps);

String POST\_URL = ps;

otc.writeBytes(sendPOST(POST\_URL)+'\n');

System.out.println("[INFO] : POST Request successful");

break;

}

ss.close();

}

}

Client side:

import java.io.\*;

import java.net.\*;

public class HTTPPostClient {

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

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

Socket clientSocket=new Socket("localhost",6789);

DataOutputStream ots=new DataOutputStream(clientSocket.getOutputStream());

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

System.out.print("[SYSTEM] > Enter URL : ");

String sentence =ifu.readLine();

ots.writeBytes(sentence+'\n');

String ms=ifs.readLine();

System.out.println("[SERVER] : POST Response :\n"+ms);

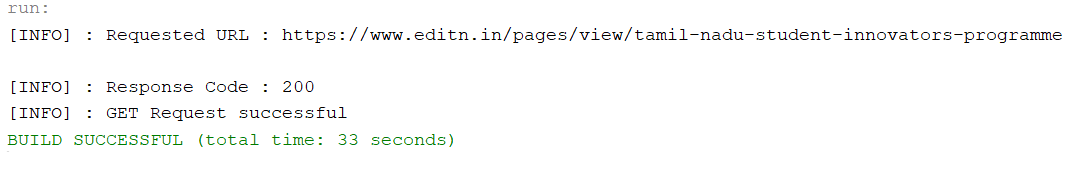
clientSocket.close();

}

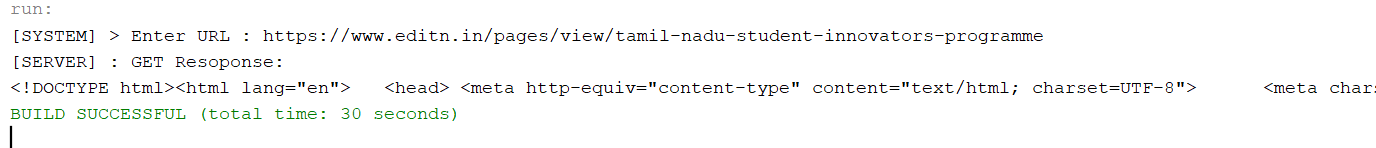
}

**Output:**

Server side:



Client side:



**Result:**

Hence, programs implementing HTTP get nd post protocol were created and executed successfully.

|  |  |
| --- | --- |
| **Ex. No: 5** | **FILE TRANSFER PROTOCOL** |
| **Date: 26/09/2022** |

**Aim:**

To create java program that implements FTP by facilitating file transfer between a server and client.

**Algorithm:**

Server side:

Client side:

**Source code:**

Server side:

import java.net.\*;

import java.io.\*;

public class FTPServer {

Socket ss;

ServerSocket serverSoc;

DataInputStream din;

DataOutputStream dout;

String fileName;

File fileObject;

public FTPServer() throws Exception{

serverSoc = new ServerSocket(6060);

System.out.println("[INFO] : Server started at "+serverSoc.getInetAddress());

ss = serverSoc.accept();

System.out.println("[INFO] : Socket connected to server from "+ss.getInetAddress());

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

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

}

public void startTransfer() throws Exception{

int readByte;

FileInputStream fStream = new FileInputStream(fileName);

System.out.println("[INFO] : Requested file tranfer to "+ss.getInetAddress()+" starting");

while((readByte = fStream.read()) != -1) dout.writeUTF(String.valueOf(readByte));

dout.writeUTF("-1");

System.out.println("[INFO] : Requested file tranfer to "+ss.getInetAddress()+" completed successfully");

fStream.close();

}

public void sendFile() throws Exception{

fileName = din.readUTF();

System.out.println("[INFO] : Requested File name: "+fileName);

fileObject = new File(fileName);

if(fileObject.exists()){

dout.writeUTF("1");

this.startTransfer();

}

else{

System.out.println("[ERROR] : Requested File not found.");

dout.writeUTF("0");

}

}

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

FTPServer instance = new FTPServer();

instance.sendFile();

instance.ss.close();

instance.serverSoc.close();

System.out.println("[INFO] : Socket disconnected and Server closed");

}

}

Client side:

import java.io.\*;

import java.net.\*;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.Scanner;;

public class FTPClient {

Socket soc;

DataInputStream din;

DataOutputStream dout;

String fileName;

public FTPClient() throws Exception{

soc = new Socket("127.0.0.1",6060);

System.out.println("[INFO] : Socket connected at "+soc.getInetAddress());

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

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

}

public void getFile() throws Exception{

String fileIn;

Scanner in = new Scanner(System.in);

System.out.print("[SYSTEM] > Enter File name to download: ");

fileName = in.nextLine();

dout.writeUTF(fileName);

Path file = Paths.get(fileName);

fileName = file.getFileName().toString();

if(din.readUTF().equals("1")) {

System.out.println("[INFO] : The Requested file is available at "+soc.getInetAddress());

File fileObj = new File(fileName);

if(fileObj.exists()){

System.out.println("[INFO] : The Requested file already existing in current folder");

System.out.print("[SYSTEM] > Do you want to replace the file[Y/N] :");

String ch = in.nextLine();

if(ch.equalsIgnoreCase("n")){

System.out.print("[SYSTEM] > Do you want to rename the recieved file[Y/N] :");

ch = in.nextLine();

if(ch.equalsIgnoreCase("n")){

System.out.println("[INFO] : File already exists. No further operations done");

in.close();

return;

}

System.out.print("[SYSTEM] > Enter new file name :");

fileName = in.nextLine();

}

}

System.out.println("[INFO] : File transfer starting from "+soc.getInetAddress());

FileOutputStream fout = new FileOutputStream(fileName);

while(!( fileIn = din.readUTF()).equals("-1")) fout.write(Integer.parseInt(fileIn));

System.out.println("[INFO] : File transfer completed from "+soc.getInetAddress());

fout.close();

}

else{

System.out.println("[ERROR] : The requested file was not found on the server");

System.out.println("[ERROR] : Check file name or try again");

}

in.close();

}

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

FTPClient instance = new FTPClient();

instance.getFile();

instance.soc.close();

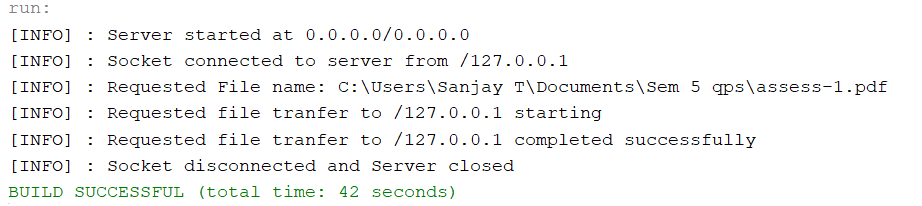
System.out.println("[INFO] : Socket closed");

}

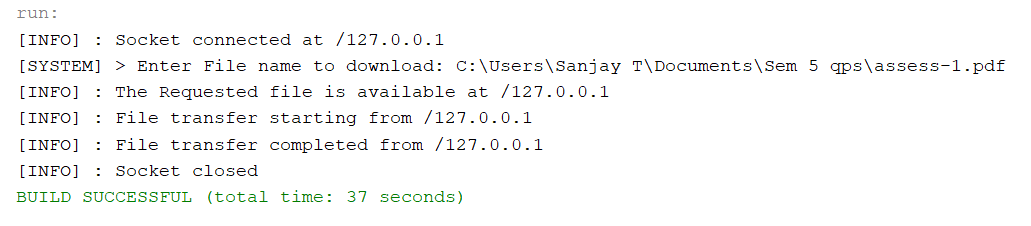
}

**Output:**

Server side:



Client side:



**Result:**

Hence, FTP was implemented using java and file transfer from server to client was performed.

|  |  |
| --- | --- |
| **Ex. No: 6** | **DOMAIN NAME SERVICE** |
| **Date: 10/10/2022** |

**Aim:**

To create java program that implements DNS Service by facilitating hostname mapping to IP address.

**Algorithm:**

Server side:

Client side:

**Source code:**

Server side:

import java.util.\*;

import java.net.\*;

public class DNS\_Server {

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();

//system.out.println(s);

InetAddress addr=receiver.getAddress();

int port=receiver.getPort();

String ip[]={"165.165.80.80","165.165.79.1","208.113.187.177"};

String name[]={"www.aptitude source.com","www.Sharifguys.com","www.mitindia.edu"};

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

{

if(s.equals(ip[i]))

{

sendbyte=name[i].getBytes();

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

server.send(sender);

break;

}

else if(s.equals(name[i]))

{

sendbyte=ip[i].getBytes();

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

server.send(sender);

break;

}

}

break;

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

Client side:

import java.io.\*;

import java.util.\*;

import java.net.\*;

public class DNS\_Client {

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 DOMAIN NAME OR IP address");

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("IP adddress or DOMAIN NAME :"+s.trim());

client.close();

}

catch(Exception e)

{

System.out.println(e);

}

}

}

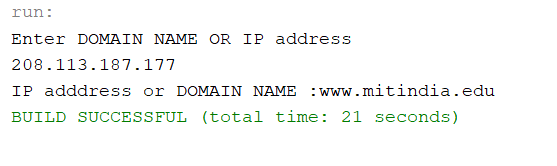
**Output:**

Server side:

A picture containing graphical user interface

Description automatically generated

Client side:



**Result:**

Hence, DNS Service is implemented from which hostname to IP address mapping is done.

|  |  |
| --- | --- |
| **Ex. No: 7** | **SIMPLE MAIL TRANSFER PROTOCOL** |
| **Date: 7/11/2022** |

**Aim:**

To create and implement simple mail transfer protocol using google smtp server.

1. **SMTP - Server :**

**Algorithm:**

**Source Code :**

import javax.mail.\*;

import javax.mail.internet.\*;

import java.util.Properties;

import java.util.Scanner;

import javax.activation.\*;

public class SendMail {

public static void main(String[] args){

Scanner in = new Scanner(System.in);

String SenderUser = "sanjayt.ifortis";

String SenderMail = "sanjayt.ifortis@gmail.com";

String SenderPassword = "aeomqwfmhphyzoa";

System.out.print("Enter Destination Address:");

String ToMail = in.nextLine();

String ToHost = "smtp.gmail.com";

Properties SessionProperties = new Properties();

SessionProperties.put("mail.smtp.auth","true");

SessionProperties.put("mail.smtp.starttls.enable","true");

SessionProperties.put("mail.smtp.host",ToHost);

SessionProperties.put("mail.smtp.port",587);

Session CurrentSession = Session.getInstance(SessionProperties,

new javax.mail.Authenticator(){

protected PasswordAuthentication getPasswordAuthentication(){

return new PasswordAuthentication(SenderMail, SenderPassword);

}

});

try{

Message ThisMessage = new MimeMessage(CurrentSession);

ThisMessage.setFrom(new InternetAddress(SenderMail));

ThisMessage.setRecipients(Message.RecipientType.TO, InternetAddress.parse(ToMail));

System.out.print("Enter Subject for mail:");

String Subject = in.nextLine();

System.out.println("Enter Body of the mail:");

String Body = in.nextLine();

ThisMessage.setSubject(Subject);

ThisMessage.setContent(Body,"text/html");

Transport.send(ThisMessage);

System.out.println("The Message was sent successfully...");

}

catch(Exception e){

e.printStackTrace();

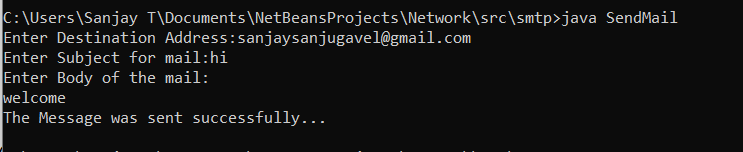
}

in.close();

}

}

**Output:**

****

1. **SMTP - HTML EMAIL :**

**Algorithm:**

**Source Code :**

import java.util.Properties;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

public class sendHTML {

public static void main(String[] args) {

// Recipient's email ID needs to be mentioned.

String to = "sanjayt.ifortis@gmail.com";

// Sender's email ID needs to be mentioned

String from = "sanjaysanjugavel@gmail.com";

// Assuming you are sending email from through gmails smtp

String host = "smtp.gmail.com";

// Get system properties

Properties properties = System.getProperties();

// Setup mail server

properties.put("mail.smtp.host", host);

properties.put("mail.smtp.port", "465");

properties.put("mail.smtp.ssl.enable", "true");

properties.put("mail.smtp.auth", "true");

// Get the Session object.// and pass username and password

Session session = Session.getInstance(properties, new javax.mail.Authenticator() {

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication("sanjayt.ifortis@gmail.com", "aeomqwfmhpjhyzoa");

}

});

// Used to debug SMTP issues

session.setDebug(true);

try {

// Create a default MimeMessage object.

MimeMessage message = new MimeMessage(session);

// Set From: header field of the header.

message.setFrom(new InternetAddress(from));

// Set To: header field of the header.

message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));

// Set Subject: header field

message.setSubject("This is the Subject Line!");

// Now set the actual message

message.setContent(

"<h1>This is actual message embedded in HTML tags</h1>",

"text/html");

//message.setText("This is actual message");

System.out.println("sending...");

// Send message

Transport.send(message);

System.out.println("Sent message successfully....");

} catch (MessagingException mex) {

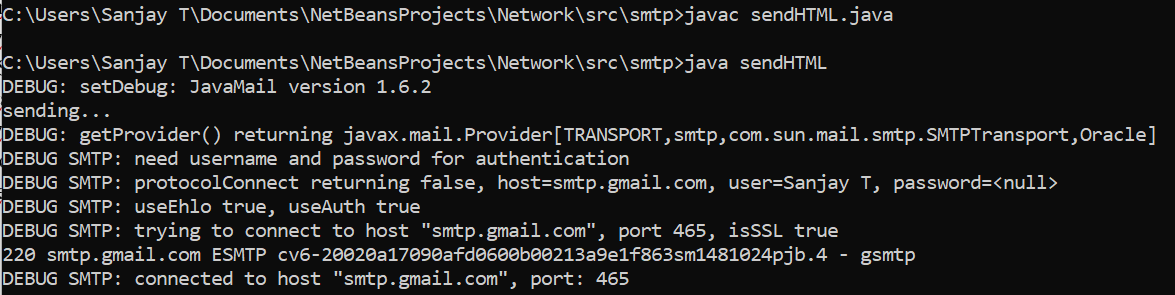
mex.printStackTrace();

}

}

}

**Output:**

****

****

**Result :**

Hence, SMTP Service is implemented from which mail is transferred from given mail to destination email when password authentication is given.

|  |  |
| --- | --- |
| **Ex. No: 8** | **POST OFFICE PROTOCOL (POP3)** |
| **Date: 8/11/2022** |

**Aim:**

To implement a pop3 protocol by check the list of mails in inbox of the mail and fetching the contents of mails provided the username and the password.

.

1. **POP SERVER – CHECK MAIL :**

**Algorithm:**

**Source Code :**

import java.util.Properties;

import javax.mail.Folder;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.NoSuchProviderException;

import javax.mail.Session;

import javax.mail.Store;

public class CheckingMails {

public static void check(String host, String storeType, String user,

String password)

{

try {

//create properties field

Properties properties = new Properties();

properties.put("mail.pop3.host", host);

properties.put("mail.pop3.port", "995");

properties.put("mail.pop3.starttls.enable", "true");

Session emailSession = Session.getDefaultInstance(properties);

//create the POP3 store object and connect with the pop server

Store store = emailSession.getStore("pop3s");

store.connect(host, user, password);

//create the folder object and open it

Folder emailFolder = store.getFolder("INBOX");

emailFolder.open(Folder.READ\_ONLY);

// retrieve the messages from the folder in an array and print it

Message[] messages = emailFolder.getMessages();

System.out.println("messages.length---" + messages.length);

for (int i = 0, n = messages.length; i < n; i++) {

Message message = messages[i];

System.out.println("---------------------------------");

System.out.println("Email Number " + (i + 1));

System.out.println("Subject: " + message.getSubject());

System.out.println("From: " + message.getFrom()[0]);

System.out.println("Text: " + message.getContent().toString());

}

//close the store and folder objects

emailFolder.close(false);

store.close();

} catch (NoSuchProviderException e) {

e.printStackTrace();

} catch (MessagingException e) {

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

String host = "pop.gmail.com";// change accordingly

String mailStoreType = "pop3";

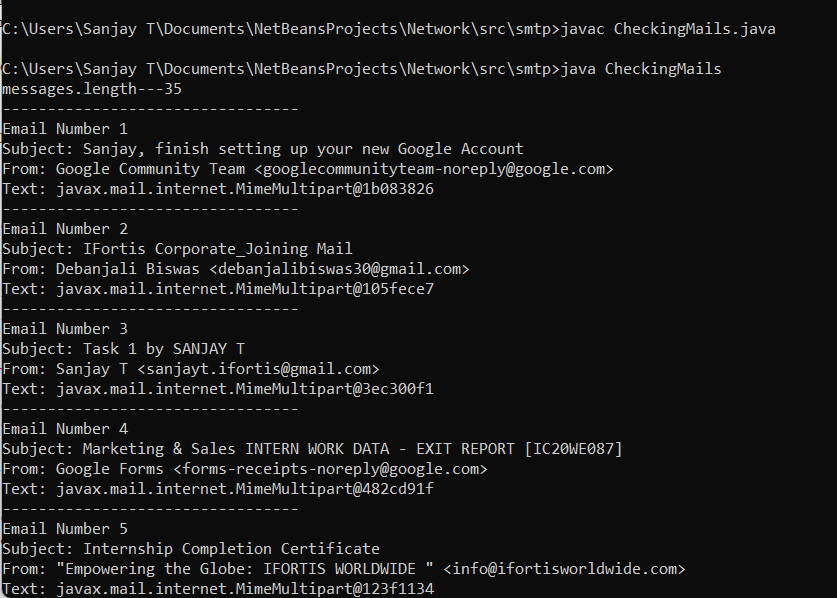
String username = "sanjayt.ifortis@gmail.com";// change accordingly

String password = "aeomqwfmhpjhyzoa";// change accordingly

check(host, mailStoreType, username, password);

}

**Output:**

****

**2) POP SERVER – FETCH MAIL :**

**Algorithm:**

**Source Code :**

import java.io.BufferedOutputStream;

import java.io.BufferedReader;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.Date;

import java.util.Properties;

import javax.mail.Address;

import javax.mail.Folder;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.Multipart;

import javax.mail.NoSuchProviderException;

import javax.mail.Part;

import javax.mail.Session;

import javax.mail.Store;

public class FetchingEmail {

public static void fetch(String pop3Host, String storeType, String user,

String password) {

try {

// create properties field

Properties properties = new Properties();

properties.put("mail.store.protocol", "pop3");

properties.put("mail.pop3.host", pop3Host);

properties.put("mail.pop3.port", "995");

properties.put("mail.pop3.starttls.enable", "true");

Session emailSession = Session.getDefaultInstance(properties);

// emailSession.setDebug(true);

// create the POP3 store object and connect with the pop server

Store store = emailSession.getStore("pop3s");

store.connect(pop3Host, user, password);

// create the folder object and open it

Folder emailFolder = store.getFolder("INBOX");

emailFolder.open(Folder.READ\_ONLY);

BufferedReader reader = new BufferedReader(new InputStreamReader(

System.in));

// retrieve the messages from the folder in an array and print it

Message[] messages = emailFolder.getMessages();

System.out.println("messages.length---" + messages.length);

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

Message message = messages[i];

System.out.println("---------------------------------");

writePart(message);

String line = reader.readLine();

if ("YES".equals(line)) {

message.writeTo(System.out);

} else if ("QUIT".equals(line)) {

break;

}

}

// close the store and folder objects

emailFolder.close(false);

store.close();

} catch (NoSuchProviderException e) {

e.printStackTrace();

} catch (MessagingException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

String host = "pop.gmail.com";// change accordingly

String mailStoreType = "pop3";

String username =

"sanjayt.ifortis@gmail.com";// change accordingly

String password = "aeomqwfmhpjhyzoa";// change accordingly

//Call method fetch

fetch(host, mailStoreType, username, password);

}

/\*

\* This method checks for content-type

\* based on which, it processes and

\* fetches the content of the message

\*/

public static void writePart(Part p) throws Exception {

if (p instanceof Message)

//Call methos writeEnvelope

writeEnvelope((Message) p);

System.out.println("----------------------------");

System.out.println("CONTENT-TYPE: " + p.getContentType());

//check if the content is plain text

if (p.isMimeType("text/plain")) {

System.out.println("This is plain text");

System.out.println("---------------------------");

System.out.println((String) p.getContent());

}

//check if the content has attachment

else if (p.isMimeType("multipart/\*")) {

System.out.println("This is a Multipart");

System.out.println("---------------------------");

Multipart mp = (Multipart) p.getContent();

int count = mp.getCount();

for (int i = 0; i < count; i++)

writePart(mp.getBodyPart(i));

}

//check if the content is a nested message

else if (p.isMimeType("message/rfc822")) {

System.out.println("This is a Nested Message");

System.out.println("---------------------------");

writePart((Part) p.getContent());

}

//check if the content is an inline image

else if (p.isMimeType("image/jpeg")) {

System.out.println("--------> image/jpeg");

Object o = p.getContent();

InputStream x = (InputStream) o;

// Construct the required byte array

System.out.println("x.length = " + x.available());

int i = 0;

byte[] bArray = new byte[x.available()];

while ((i = (int) ((InputStream) x).available()) > 0) {

int result = (int) (((InputStream) x).read(bArray));

if (result == -1)

break;

}

FileOutputStream f2 = new FileOutputStream("/tmp/image.jpg");

f2.write(bArray);

}

else if (p.getContentType().contains("image/")) {

System.out.println("content type" + p.getContentType());

File f = new File("image" + new Date().getTime() + ".jpg");

DataOutputStream output = new DataOutputStream(

new BufferedOutputStream(new FileOutputStream(f)));

com.sun.mail.util.BASE64DecoderStream test =

(com.sun.mail.util.BASE64DecoderStream) p

.getContent();

byte[] buffer = new byte[1024];

int bytesRead;

while ((bytesRead = test.read(buffer)) != -1) {

output.write(buffer, 0, bytesRead);

}

}

else {

Object o = p.getContent();

if (o instanceof String) {

System.out.println("This is a string");

System.out.println("---------------------------");

System.out.println((String) o);

}

else if (o instanceof InputStream) {

System.out.println("This is just an input stream");

System.out.println("---------------------------");

InputStream is = (InputStream) o;

is = (InputStream) o;

int c;

while ((c = is.read()) != -1)

System.out.write(c);

}

else {

System.out.println("This is an unknown type");

System.out.println("---------------------------");

System.out.println(o.toString());

}

}

}

/\*

\* This method would print FROM,TO and SUBJECT of the message

\*/

public static void writeEnvelope(Message m) throws Exception {

System.out.println("This is the message envelope");

System.out.println("---------------------------");

Address[] a;

// FROM

if ((a = m.getFrom()) != null) {

for (int j = 0; j < a.length; j++)

System.out.println("FROM: " + a[j].toString());

}

// TO

if ((a = m.getRecipients(Message.RecipientType.TO)) != null) {

for (int j = 0; j < a.length; j++)

System.out.println("TO: " + a[j].toString());

}

// SUBJECT

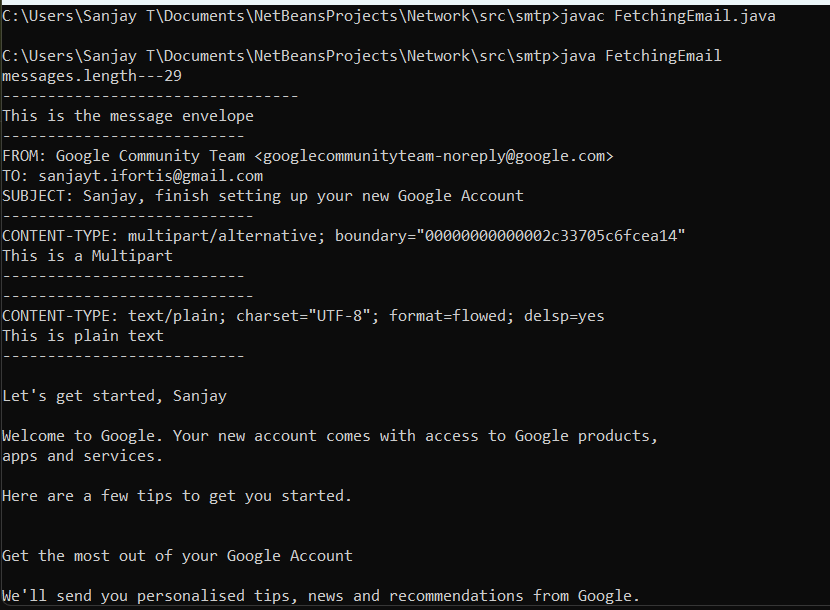
if (m.getSubject() != null)

System.out.println("SUBJECT: " + m.getSubject());

}

}

**Output:**



**Result :**

Thus, checking and fetching of mail from inbox is done using POP Server and is implemented in java.

|  |  |
| --- | --- |
| **Ex. No: 9** | **PING COMMAND USING SOCKET** |
| **Date: 14/11/2022** |

**Aim:**

To implement ping client server program in java using sockets

**Algorithm:**

Server side:

Client side:

**Source code:**

Server side:

:

package ping;

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class PingServer

{

public static void main(String[] args)

{

try

{

ServerSocket ss=new ServerSocket(2156);

Socket s=ss.accept();

if(s.isConnected())

System.out.println("Connected ...");

System.out.println("Listening ...");

DataOutputStream dos;

try (DataInputStream dis = new DataInputStream(s.getInputStream())) {

dos = new DataOutputStream(s.getOutputStream());

int no=0;

String ip="";

if((dis.readUTF()).equals("P"))

{

System.out.println("Getting No. Of Packets ...");

no=dis.readInt();

} if((dis.readUTF()).equals("A"))

{

System.out.println("Getting the Address ...");

ip=dis.readUTF();

} Process p=Runtime.getRuntime().exec("ping -n "+no+" "+ip);

System.out.println("Running ping -n "+no+" "+ip);

BufferedReader br=new BufferedReader(new InputStreamReader(p.getInputStream()));

String ipline=br.readLine();

while(ipline != null )

{

dos.writeUTF(ipline);

ipline=br.readLine();

} }

dos.close();

}

catch(Exception x)

{

x.printStackTrace();

}

}

}

Client side:

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class PingClient

{

public static void main(String[] args)

{

try

{

Socket s=new Socket("localhost",2156);

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

if(s.isConnected())

System.out.println("Connected !!");

Scanner in=new Scanner(System.in);

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

DataOutputStream os=new DataOutputStream(s.getOutputStream());

System.out.println("How many Packets You want to send ? ");

int no=in.nextInt();

System.out.println("Address to be pinged :");

String ip=br.readLine();

os.writeUTF("P");

os.writeInt(no);

os.writeUTF("A");

os.writeUTF(ip);

String pingline=is.readUTF();

while(pingline != null )

{

System.out.println(pingline);

pingline=is.readUTF();

}

os.flush();

os.close();

is.close();

}catch(Exception x)

{

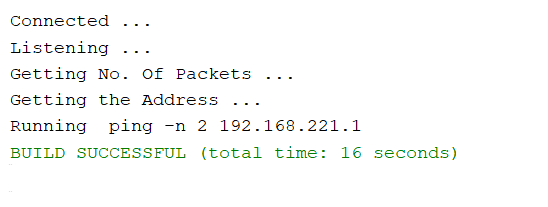
}

}

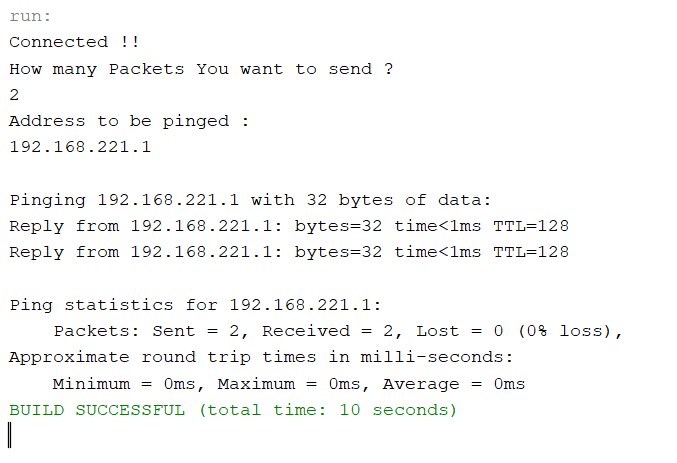
}

**Output :**

Server side:



Client side:



**Result :**

Thus, PING command is implemented using sockets in java programming .

|  |  |
| --- | --- |
| **Ex. No: 10** | **RIP AND OSPF PROTOCOLS** |
| **Date: 14/11/2022** |

**ROUTING INFORMATION PROTOCOL**

**Aim:**

To simulate Routing Information Protocol using Cisco Packet Tracer.

**Procedure :**

1.Download the Cisco Packet Tracer and Login, then open a new Window.

2.Select two End Devices(2 PC’S) and Drag them from the menu.

3.Select two Network Devices(PT routers) and two PT - Switches and Drag them from the menu.

4.Connect All of them as shown in the output.

5.Name the two networks as A and B and assign them a particular IP address.

6.Configure the two End Devices using the IP address assigned.

7.Similarly assign the IP address between the two connected Routers.

8.Send the Packet PC0 to Router0 and Similarly between PC1 to Router1.

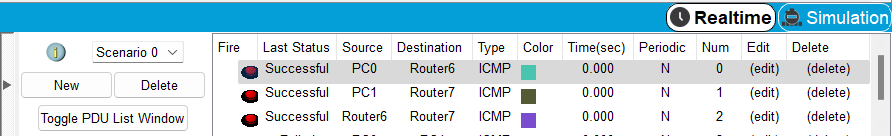
9.Now try to send the packet between PC0and PC1,if it fails go to step 10.

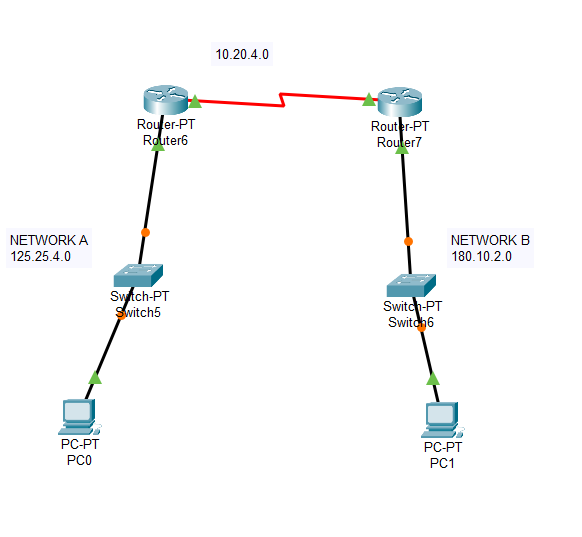
10.Configure the Routers and add all the IP address to the RIP of Routers and save them.

11.Now send the packets from PC0 to PC1 and from Router0 to Router1.

12.The Packets are being successfully sent.

**Output :**

****



**Result :**

Hence Routing Information Protocol is simulated and the packet is successfully sent from the

PC0 to PC1 and vice versa.

**OPEN SHORTEST PATH FIRST PROTOCOL (OSPF)**

**Aim :**

To simulate the Open Shortest Path First Protocol (OSPF) in Cisco Packet Tracer.

**Procedure :**

1.Login Cisco Packet Tracer and Open a new Window.

2.Drag in two PC’S from the End Devices.

3.Drag in Three Routers and Connect all the Devices.

4.Name the networks with appropriate IP address.

5.Configure the PC’S to their respective networks.

6.Connect the Routers in their appropriate Networks.

7.Now use the OSPF to make all the networks visible to each other.

8.See the Serial Routers are Connected it and Configure them.Use the Command line Interpreter to activate the OSPF.

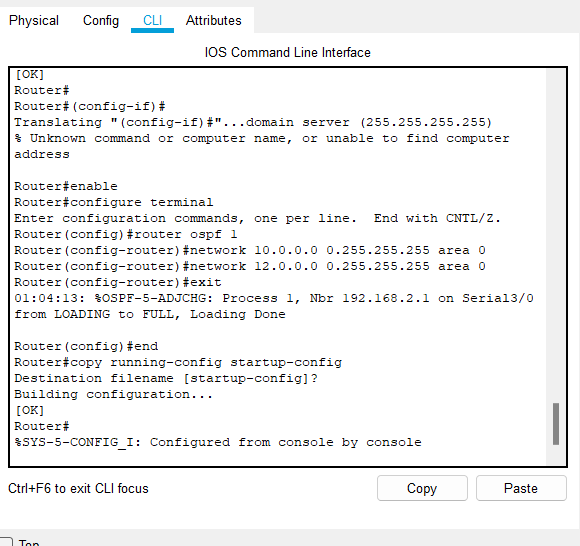
9.Enable and Configure the terminal in the CLI.

10.Now Connect the Router to the Networks it is Connected.

11.The OSPF is activated hence making all the networks visible to each other.

12.Now send a Packet from PC0 to PC1 and Vice versa, the Routers choose the path to deliver the packet.

**Output :** Router OSPF Configuration using CLI

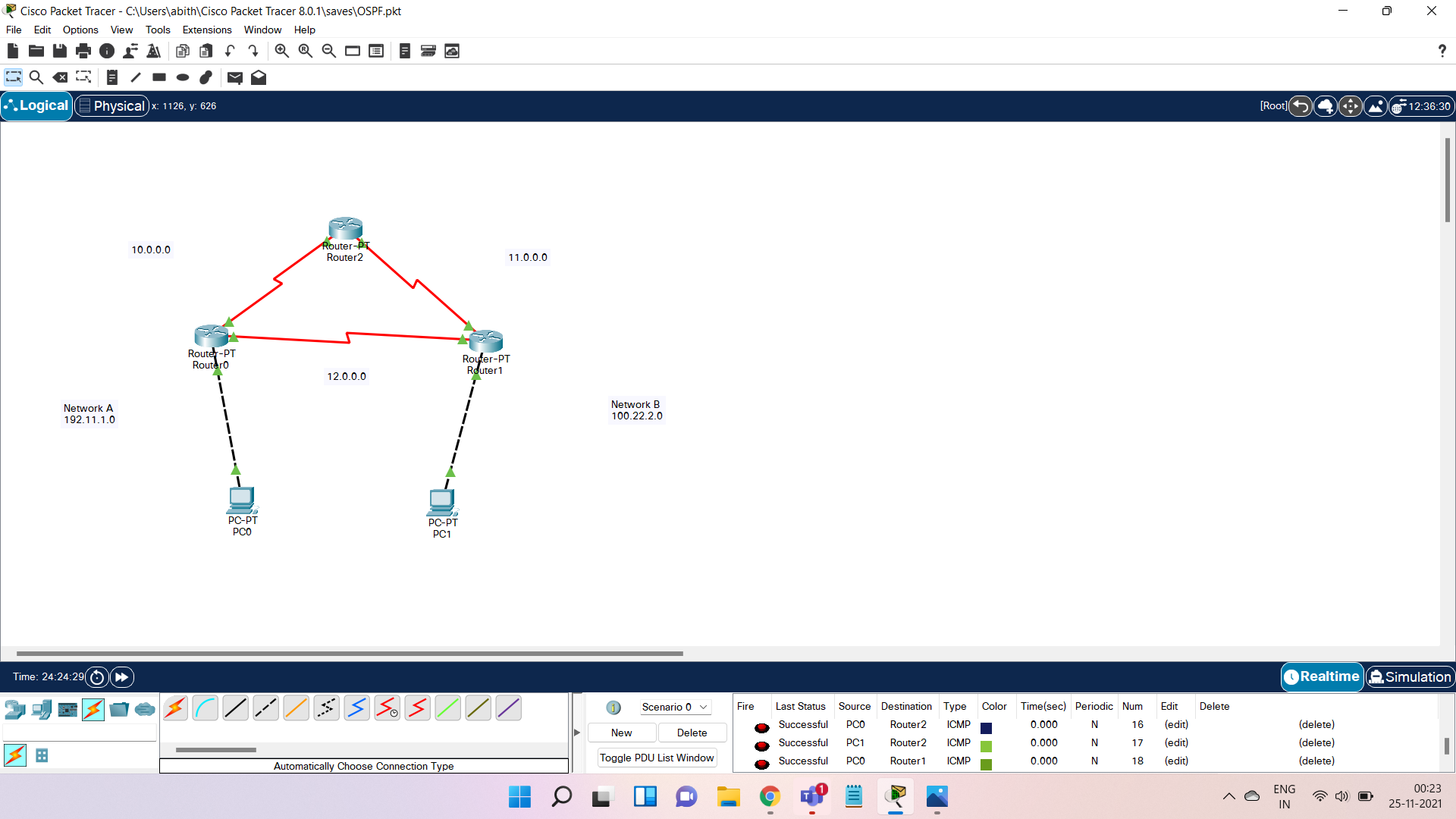


Graphical user interface, text, application

Description automatically generated

Diagram

Description automatically generated



**Result :**

Hence Open Shortest Path First (OSPF) is simulated and the packet is successfully sent from the PC0

to PC1 and vice versa.

|  |  |
| --- | --- |
| **Ex. No: 11** | **STUDY ON NETWORK SIMULATOR (NS2)** |
| **Date: 14/11/2022** |

**Aim :**

To study network simulator version 2 (ns2)

|  |  |
| --- | --- |
| **Ex. No: 12** | **PING COMMAND USING SOCKET** |
| **Date: 14/11/2022** |