**Write a code simulating ARP /RARP protocols**

**Write a code simulating ARP /RARP protocols.**

**Aim:**

       To write a java program for simulating ARP/RARP protocols

**ALGORITHM:**

**server**

1. Create a server socket and bind it to port.

2. Listen for new connection and when a connection arrives, accept it.

3. Send server‟s date and time to the client.

4. Read client‟s IP address sent by the client.

5. Display the client details.

6. Repeat steps 2-5 until the server is terminated.

7. Close all streams.

8. Close the server socket.

9. Stop.

**Client**

1. Create a client socket and connect it to the server‟s port number.

2. Retrieve its own IP address using built-in function.

3. Send its address to the server.

4. Display the date & time sent by the server.

5. Close the input and output streams.

6. Close the client socket.

7. Stop.

Program

Program for Address Resolutuion Protocol (ARP) using TCP

**Client:**

import java.io.\*;

import java.net.\*;

import java.util.\*;

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",139);

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);

}

}

}

**Server:**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Serverarp

{

public static void main(String args[])

{

try

{

ServerSocket obj=new ServerSocket(139);

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"};

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

{

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

{

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

break;

}

}

obj.close();

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**Client:**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Clientrarp12

{

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);

}

}

}

**Server:**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Serverrarp12

{

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"};

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

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);

break;

}

}

break;

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**EX-NO:2. Applications using TCP Sockets like**

**a. Echo client and echo server**

**Aim**

        To write a java program for appalaction using TCP Sockets Links

**Algorithm**

1.Start the program.

2.Get the frame size from the user

3.To create the frame based on the user request.

4.To send frames to server from the client side.

5.If your frames reach the server it will send ACK signal to client     otherwise it will

send NACK signal to client.

6.Stop the program

**Program :**

**/echo client.java**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class echoclient

{

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

{

Socket c=null;

DataInputStream usr\_inp=null;

DataInputStream din=new DataInputStream(System.in);

DataOutputStream dout=null;

try

{

c=new Socket("127.0.0.1",5678);

usr\_inp=new DataInputStream(c.getInputStream());

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

}

catch(IOException e)

{

}

www.studentsfocus.com

if(c!=null || usr\_inp!=null || dout!=null)

{

String unip;

while((unip=din.readLine())!=null)

{

dout.writeBytes(""+unip);

dout.writeBytes("\n");

System.out.println("\n the echoed message");

System.out.println(usr\_inp.readLine());

System.out.println("\n enter your message");

}

System.exit(0);

}

din.close();

usr\_inp.close();

c.close();

}

}

**//echoserver.java**

import java.io.\*;

import java.net.\*;

public class echoserver

{

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

{

ServerSocket m=null;

Socket c=null;

DataInputStream usr\_inp=null;

DataInputStream din=new DataInputStream(System.in);

DataOutputStream dout=null;

try

{

m=new ServerSocket(5678);

c=m.accept();

usr\_inp=new DataInputStream(c.getInputStream());

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

}

catch(IOException e)

{}

if(c!=null || usr\_inp!=null)

www.studentsfocus.com

{

String unip;

while(true)

{

System.out.println("\nMessage from Client...");

String m1=(usr\_inp.readLine());

System.out.println(m1);

dout.writeBytes(""+m1);

dout.writeBytes("\n");

}

}

dout.close();

usr\_inp.close();

c.close();

}

}

**b. Chat**

/talkclient.java

import java.io.\*;

import java.net.\*;

public class talkclient

{

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

{

Socket c=null;

DataInputStream usr\_inp=null;

DataInputStream din=new DataInputStream(System.in);

DataOutputStream dout=null;

try

www.studentsfocus.com

{

c=new Socket("127.0.0.1",1234);

usr\_inp=new DataInputStream(c.getInputStream());

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

}

catch(IOException e)

{}

if(c!=null || usr\_inp!=null || dout!=null)

{

String unip;

System.out.println("\nEnter the message for server:");

while((unip=din.readLine())!=null)

{

dout.writeBytes(""+unip);

dout.writeBytes("\n");

System.out.println("reply");

System.out.println(usr\_inp.readLine());

System.out.println("\n enter your message:");

}

System.exit(0);

}

din.close();

usr\_inp.close();

c.close();

}

}

**//talkserver.java**

import java.io.\*;

import java.net.\*;

public class talkserver

{

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

{

ServerSocket m=null;

Socket c=null;

DataInputStream usr\_inp=null;

DataInputStream din=new DataInputStream(System.in);

DataOutputStream dout=null;

try

{

www.studentsfocus.com

m=new ServerSocket(1234);

c=m.accept();

usr\_inp=new DataInputStream(c.getInputStream());

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

}

catch(IOException e)

{}

if(c!=null||usr\_inp!=null)

{

String unip;

while(true)

{

System.out.println("\nmessage from client:");

String m1=usr\_inp.readLine();

System.out.println(m1);

System.out.println("enter your message:");

unip=din.readLine();

dout.writeBytes(""+unip);

dout.writeBytes("\n");

}

}

dout.close();

usr\_inp.close();

c.close();

}

}

**C. File Transfer**

**Program**

**File Client**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Clientfile

{ public static void main(String args[])

{

Try

{

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

Socket clsct=new Socket("127.0.0.1",139);

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

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

System.out.println("Enter the file name:");

String str=in.readLine();

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

System.out.println("Enter the new file name:");

String str2=in.readLine();

String str1,ss;

FileWriter f=new FileWriter(str2);

char buffer[];

while(true)

{ str1=din.readLine();

if(str1.equals("-1")) break;

System.out.println(str1);

buffer=new char[str1.length()];

str1.getChars(0,str1.length(),buffer,0);

f.write(buffer);

}

f.close();

clsct.close();

}

catch (Exception e)

{

System.out.println(e);

}

}

}

**Server**

import java.io.\*;

import java.net.\*;

import java.util.\*;

class Serverfile

{ public static void main(String args[])

{

Try

{

ServerSocket obj=new ServerSocket(139);

while(true)

{

Socket obj1=obj.accept();

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

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

String str=din.readLine();

FileReader f=new FileReader(str);

BufferedReader b=new BufferedReader(f);

String s;

while((s=b.readLine())!=null)

{ System.out.println(s);

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

}

f.close();

dout.writeBytes("-1\n");

} }

catch(Exception e)

{ System.out.println(e);}

}

}

**EX-NO 3. Study of Network simulator (NS).and Simulation of Congestion Control Algorithms using NS**

**Aim:**

To Study of Network simulator (NS).and Simulation of Congestion Control Algorithms using NS

**NET WORK SIMULATOR (NS2)**

**Ns overview**

Ø  Ns programming: A Quick start

Ø Case study I: A simple Wireless network

Ø Case study II: Create a new agent in Ns

Ns overview

Ø Ns Status

Ø Periodical release (ns-2.26, Feb 2003)

Ø Platform support

Ø FreeBSD, Linux, Solaris, Windows and Mac

**Ns unctionalities**

            Routing, Transportation, Traffic sources,Queuing

            disciplines, QoS

**Wireless**

           Ad hoc routing, mobile IP, sensor-MAC

           Tracing, visualization and various utilitie

            NS(Network Simulators)

            Most of the commercial simulators are GUI driven, while some network simulators areCLI driven. The network model / configuration describes the state of the network (nodes,routers,switches, links) and the events (data transmissions, packet error etc.). An important output ofsimulations are 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 ofnotable 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 the users to 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 thenodes. 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**

occurs when one or morepacketsof data travelling across a computer networkfail to reachtheir destination. Packet loss is distinguished as one of the three main error types encountered in digital communications; the other two being bit errorand 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 transportlayer protocols to maintain high bandwidths, the sensitivity to loss of individual packets, as well as to frequency and patterns of loss among longer packet sequences is strongly dependent on the application itself.

**Throughput**

This is the main performance measure characteristic, and most widely used. Incommunicationnetworks, such asEthernetorpacket radio, throughputor network throughputis the average rate of successfulmessage delivery over a communication channel. Thethroughput is usually measured inbitsper second (bit/s orbps), andsometimes indata packetsper second or data packets pertime slotThis measure how soon the receiver is able to get a certain amount of data send 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 which directly impacts the 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 valueof delay, the more difficult it is for transport layer protocols to maintain highbandwidths. 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 thesystem after being served. Thus queue length is very important characteristic to determine that how well the activequeue management of the congestion control algorithm has been working.

**RESULT**

          Thus the study of Network simulator (NS2)was studied