**Ex.No: 2 Write a HTTP web client program to download a web page using TCP sockets**

**AIM:**

To write a java program for socket for HTTP for web page upload and download .

**ALGORITHM:**

**Client:**

1. Start.

2. Create socket and establish the connection with the server.

3. Read the image to be uploaded from the disk

4. Send the image read to the server

5. Terminate the connection

6. Stop.

**Server:**

1. Start

2. Create socket, bind IP address and port number with the created socket and make

server a listening server.

3. Accept the connection request from the client

4. Receive the image sent by the client.

5. Display the image.

6. Close the connection.

7. Stop.

**PROGRAM**

**Client**

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 Client

{

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

{

Socket soc;

BufferedImage img = null;

soc=new

Socket(“localhost”,4000);

System.out.println(“Client is running.

“):

try {

System.out.println(“Reading image from disk.”);

img = ImageIO.read(new File(“digital\_image\_processing.jpg”));

ByteArrayOutputStream baos = new ByteArrayOutputStream();

ImageIO.write(img, “jpg”, baos);

baos.flush();

byte[] bytes = baos.toByteArray(); baos.close();

System.out.println(“Sending image to 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 server. “);

dos.close();

out.close();

}

catch (Exception e)

{

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

soc.close();

}

soc.close();

}

}

**Server**

import java.net.\*;

import java.io.\*;

import java.awt.image.\*;

import javax.imageio.\*;

import javax.swing.\*;

class Server

{

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

{

ServerSocket server=null;

Socket socket;

server=new ServerSocket(4000);

System.out.println(“Server Waiting for 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:**

When you run the client code, following output screen would appear on client side.

A black screen with white text

Description automatically generated

**RESULT:**

Thus the socket program for HTTP for web page upload and download was developed and

executed successfully.

**Ex.No: 3 Applications using TCP sockets like: Echo client and echo server, Chat and File Transfer**

**AIM**

To write a java program for application using TCP Sockets Links

**a.Echo client and echo server**

**ALGORITHM**

**Client**

1. Start

2. Create the TCP socket

3. Establish connection with the server

4. Get the message to be echoed from the user

5. Send the message to the server

6. Receive the message echoed by the server

7. Display the message received from the server

8. Terminate the connection

9. Stop

**Server**

1. Start

2. Create TCP socket, make it a listening socket

3. Accept the connection request sent by the client for connection establishment

4. Receive the message sent by the client

5. Display the received message

6. Send the received message to the client from which it receives

7. Close the connection when client initiates termination and server becomes a listening server, waiting for clients.

8. Stop.

**PROGRAM:**

**EchoServer.java**

import java.net.\*;

import java.io.\*;

public class EServer

{

public static void main(String args[])

{

ServerSocket s=null;

String line;

DataInputStream is;

PrintStream ps;

Socket c=null;

try

{

s=new ServerSocket(9000);

}

catch(IOException e)

{

System.out.println(e);

}

try

{

c=s.accept();

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

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

while(true)

{

line=is.readLine();

ps.println(line);

}

}

catch(IOException e)

{

System.out.println(e);

}

}

}

**EClient.java**

import java.net.\*;

import java.io.\*;

public class EClient

{ public static void main(String arg[])

{

Socket c=null;

String line;

DataInputStream is,is1;

PrintStream os;

try

{

InetAddress ia = InetAddress.getLocalHost();

c=new Socket(ia,9000);

}

catch(IOException e)

{

System.out.println(e);

}

try

{

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

is=new DataInputStream(System.in);

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

while(true)

{

System.out.println(“Client:”);

line=is.readLine();

os.println(line);

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

}

}

catch(IOException e)

{

System.out.println(“Socket Closed!”);

}

}}

**OUTPUT**

**Server**

C:\Program Files\Java\jdk1.5.0\bin>javac EServer.java

C:\Program Files\Java\jdk1.5.0\bin>java EServer

C:\Program Files\Java\jdk1.5.0\bin>

**Client**

C:\Program Files\Java\jdk1.5.0\bin>javac EClient.java

C:\Program Files\Java\jdk1.5.0\bin>java EClient

Client: Hai Server

Server:Hai Server

Client: Hello

Server:Hello

Client:end

Server:end

Client:ds

Socket Closed!

**B.Chat**

**ALGORITHM**

**Client**

1. Start

2. Create the UDP datagram socket

3. Get the request message to be sent from the user

4. Send the request message to the server

5. If the request message is “END” go to step 10

6. Wait for the reply message from the server

7. Receive the reply message sent by the server

8. Display the reply message received from the server

9. Repeat the steps from 3 to 8

10. Stop

**Server**

1. Start

2. Create UDP datagram socket, make it a listening socket

3. Receive the request message sent by the client

4. If the received message is “END” go to step 10

5. Retrieve the client’s IP address from the request message received

6. Display the received message

7. Get the reply message from the user

8. Send the reply message to the client

9. Repeat the steps from 3 to 8.

10. Stop.

**PROGRAM**

**UDPserver.java**

import java.io.\*;

import java.net.\*;

class UDPserver

{

public static DatagramSocket ds;

public static byte buffer[]=new byte[1024];

public static int clientport=789,serverport=790;

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

{

ds=new DatagramSocket(clientport);

System.out.println(“press ctrl+c to quit the program”);

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

InetAddress ia=InetAddress.geyLocalHost();

while(true)

{

DatagramPacket p=new DatagramPacket(buffer,buffer.length);

ds.receive(p);

String psx=new String(p.getData(),0,p.getLength());

System.out.println(“Client:” + psx);

System.out.println(“Server:”);

String str=dis.readLine();

if(str.equals(“end”))

break;

buffer=str.getBytes();

ds.send(new DatagramPacket(buffer,str.length(),ia,serverport));

}

}

}

**UDPclient.java**

import java .io.\*;

import java.net.\*;

class UDPclient

{

public static DatagramSocket ds;

public static int clientport=789,serverport=790;

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

{

byte buffer[]=new byte[1024];

ds=new DatagramSocket(serverport);

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

System.out.println(“server waiting”);

InetAddress ia=InetAddress.getLocalHost();

while(true)

{

System.out.println(“Client:”);

String str=dis.readLine();

if(str.equals(“end”))

break;

buffer=str.getBytes();

ds.send(new DatagramPacket(buffer,str.length(),ia,clientport));

DatagramPacket p=new DatagramPacket(buffer,buffer.length);

ds.receive(p);

String psx=new String(p.getData(),0,p.getLength());

System.out.println(“Server:” + psx);

}

}

}

**OUTPUT:**

**Server**

C:\Program Files\Java\jdk1.5.0\bin>javac UDPserver.java

C:\Program Files\Java\jdk1.5.0\bin>java UDPserver

press ctrl+c to quit the program

Client:Hai Server

Server:Hello Client

Client:How are You

Server:I am Fine

**Client**

C:\Program Files\Java\jdk1.5.0\bin>javac UDPclient.java

C:\Program Files\Java\jdk1.5.0\bin>java UDPclient

server waiting

Client:Hai Server

Server:Hello Clie

Client:How are You

Server:I am Fine

Client:end

**C. File Transfer**

**AIM:**

To write a java program for file transfer using TCP Sockets.

**Algorithm**

**Server**

1. Import java packages and create class file server.

2. Create a new server socket and bind it to the port.

3. Accept the client connection

4. Get the file name and stored into the BufferedReader.

5. Create a new object class file and realine.

6. If file is exists then FileReader read the content until EOF is reached.

7. Stop the program.

**Client**

1. Import java packages and create class file server.

2. Create a new server socket and bind it to the port.

3. Now connection is established.

4. The object of a BufferReader class is used for storing data content which has been retrieved

from socket object.

5. The connection is closed.

6. Stop the program.

**PROGRAM**

**File Server :**

import java.io.BufferedInputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.OutputStream;

import java.net.InetAddress;

import java.net.ServerSocket;

import java.net.Socket

public class FileServer

{

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

{

//Initialize Sockets

ServerSocket ssock = new ServerSocket(5000); Socket

socket = ssock.accept();

//The InetAddress specification

InetAddress IA = InetAddress.getByName(“localhost”);

//Specify the file

File file = new File(“e:\\Bookmarks.html”);

FileInputStream fis = new

FileInputStream(file);

BufferedInputStream bis = new BufferedInputStream(fis); //Get

socket’s output stream

OutputStream os = socket.getOutputStream(); //Read

File Contents into contents array

byte[] contents;

long fileLength = file.length();

long current = 0;

long start = System.nanoTime();

while(current!=fileLength){

int size = 10000;

if(fileLength - current >= size)

current += size;

else{

size = (int)(fileLength - current);

current = fileLength;

}

contents = new byte[size];

bis.read(contents, 0, size);

os.write(contents);

System.out.print(“Sending file ... “+(current\*100)/fileLength+”% complete!”);

}

os.flush();

//File transfer done. Close the socket connection!

socket.close();

ssock.close();

System.out.println(“File sent succesfully!”);

} }

**File Client:**

import java.io.BufferedOutputStream;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.net.InetAddress;

import java.net.Socket;

public class FileClient {

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

//Initialize socket

Socket socket = new Socket(InetAddress.getByName(“localhost”), 5000); byte[]

contents = new byte[10000];

//Initialize the FileOutputStream to the output file’s full path. FileOutputStream fos = new

FileOutputStream(“e:\\Bookmarks1.html”);

BufferedOutputStream bos = new

BufferedOutputStream(fos); InputStream is =

socket.getInputStream();

//No of bytes read in one read() call

int bytesRead = 0;

while((bytesRead=is.read(contents))!=-1)

bos.write(contents, 0, bytesRead);

bos.flush();

socket.close();

System.out.println(“File saved successfully!”);

}

}

**Output**

**server**

E:\nwlab>java FileServer

Sending file ... 9% complete!

Sending file ... 19% complete!

Sending file ... 28% complete!

Sending file ... 38% complete!

Sending file ... 47% complete!

Sending file ... 57% complete!

Sending file ... 66% complete!

Sending file ... 76% complete!

Sending file ... 86% complete!

Sending file ... 95% complete!

Sending file ... 100% complete!

File sent successfully!

E:\nwlab>**client**

E:\nwlab>java FileClient

File saved successfully!

E:\nwlab>

**RESULT:**

Thus the java application program using TCP Sockets was developed and executed

successfully.

**Ex.No:4 Simulation of DNS using UDP Sockets**

**AIM**

To write a java program for DNS application

**ALGORITHM**

**Server**

1. Start

2. Create UDP datagram socket

3. Create a table that maps host name and IP address

4. Receive the host name from the client

5. Retrieve the client’s IP address from the received datagram

6. Get the IP address mapped for the host name from the table.

7. Display the host name and corresponding IP address

8. Send the IP address for the requested host name to the client

9. Stop.

**Client**

1. Start

2. Create UDP datagram socket.

3. Get the host name from the client

4. Send the host name to the server

5. Wait for the reply from the server

6. Receive the reply datagram and read the IP address for the requested host name

7. Display the IP address.

8. Stop.

**PROGRAM**

**DNS Server**

java import java.io.\*;

import java.net.\*;

public class udpdnsserver

{

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

{

str = str.trim();

for (int i=0; i &lt; array.length; i++)

{

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

return i;

}

return -1;

}

public static void main(String arg[])throws IOException

{

String[] hosts = {“yahoo.com”, “gmail.com”,”cricinfo.com”, “facebook.com”};

String[] ip = {“68.180.206.184”, “209.85.148.19”,”80.168.92.140”, “69.63.189.16”};

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();

}}}

**UDP DNS Client**

java import java.io.\*;

import java.net.\*;

public class udpdnsclient

{

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,portaddr);

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,

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**

**Server**

javac udpdnsserver.java

java udpdnsserver

Press Ctrl + C to Quit Request for host yahoo.com

Request for host cricinfo.com

Request for host youtube.com

**Client**

>javac udpdnsclient.java

>java udpdnsclient

Enter the hostname : yahoo.com

IP Address: 68.180.206.184

>java udpdnsclient

Enter the hostname : cricinfo.com

IP Address: 80.168.92.140

>java udpdnsclient

Enter the hostname : youtube.com

IP Address: Host Not Found

**RESULT:**

Thus the java application program using UDP Sockets to implement DNS was developed and

executed successfully

**Ex.No:5 Write a code simulating ARP /RARP protocols**

**AIM:**

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

**ALGORITHM:**

**Client**

1. Start the program

2. Create socket and establish connection with the server.

3. Get the IP address to be converted into MAC address from the user.

4. Send this IP address to server.

5. Receive the MAC address for the IP address from the server.

6. Display the received MAC address

7. Terminate the connection

**Server**

1. Start the program

2. Create the socket, bind the socket created with IP address and port number and make it a listening socket.

3. Accept the connection request when it is requested by the client.

4. Server maintains the table in which IP and corresponding MAC addresses are

stored.

5. Receive the IP address sent by the client.

6. Retrieve the corresponding MAC address for the IP address and send it to the client.

7. Close the connection with the client and now the server becomes a listening server

waiting for the connection request from other clients

8. Stop

**PROGRAM**

**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&lt;ip.length;i++)

{

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

{

dout.writeBytes(mac[i]+’\n’);

break;

}

}

obj.close();

}

}

catch(Exception e)

{

System.out.println(e);

}}

}

**Output:**

E:\networks>java Serverarp

E:\networks>java Clientarp

Enter the Logical address(IP):

165.165.80.80

The Physical Address is: 6A:08:AA:C2

**(b) Program for Reverse Address Resolution Protocol (RARP) using UDP**

**ALGORITHM:**

**Client**

1. Start the program

2. Create datagram socket

3. Get the MAC address to be converted into IP address from the user.

4. Send this MAC address to server using UDP datagram.

5. Receive the datagram from the server and display the corresponding IP address.

6. Stop

**Server**

1. Start the program.

2. Server maintains the table in which IP and corresponding MAC addresses are

stored.

3. Create the datagram socket

4. Receive the datagram sent by the client and read the MAC address sent.

5. Retrieve the IP address for the received MAC address from the table.

6. Display the corresponding IP address.

7. Stop

**PROGRAM:**

**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();

DatagramPacketsender=newDatagramPacket(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();

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&lt;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);

}}}

**Output:**

I:\ex>java Serverrarp12

I:\ex>java Clientrarp12

Enter the Physical address (MAC):

6A:08:AA:C2

The Logical Address is(IP): 165.165.80.80

**RESULT :**

Thus the program for implementing to display simulating ARP and RARP protocols was

executed successfully and output is verified.

**Ex.No: 6 Study of Network simulator (NS) and Simulation of Congestion Control**

**Algorithms using NS**

**AIM:**

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

using NS

**ALGORITHM:**

1. Initialize pointers elm, elm1, elm2 and integer num\_later.
2. Start traversing the received history queue from the first element.
3. Continue traversal while elm is not NULL and num\_later is less than or equal to num\_dup\_acks\_.
4. If elm is found, call findDataPacketInRecvHistory to locate a data packet starting from elm.
5. Traverse from the identified data packet, removing entries with sequence numbers and received times less than specified values.
6. Define a function to traverse the history and remove entries marked as ACK.
7. Define a function to skip over ACK entries and return the next data packet.

**Program:**

include &lt;wifi\_lte/wifi\_lte\_rtable.h>

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

i nt num\_later = 1;

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

while (elm != NULL &amp;&amp; num\_later &lt;= 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\_ &lt; seq\_num &amp;&amp; elm2->t\_recv\_ &lt;

time){

STAILQ\_REMOVE(&amp;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(&amp;r\_hist\_);

struct r\_hist\_entry \*elm2;

int num\_later = 1;

while (elm1 != NULL &amp;&amp; num\_later &lt;= 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(&amp;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 &amp;&amp; start->type\_ == DCCP\_ACK)

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

return start;

}

**Result:**

Thus we have Studied Network simulator (NS) and Simulation of Congestion Control

Algorithms using NS.