/\*\*\*\*\*\* A3 CRC \*\*\*\*\*\*/

include <stdio.h>

#include <string.h>

void main() {

int i,j,keylen,msglen;

char input[100], key[30],temp[30],quot[100],rem[30],key1[30];

printf("Enter Data: ");

scanf("%s",input);

printf("Enter Key: ");

scanf("%s",key);

keylen=strlen(key);

msglen=strlen(input);

strcpy(key1,key);

for (i=0;i<keylen-1;i++) {

input[msglen+i]='0';

}

for (i=0;i<keylen;i++)

temp[i]=input[i];

for (i=0;i<msglen;i++) {

quot[i]=temp[0];

if(quot[i]=='0')

for (j=0;j<keylen;j++)

key[j]='0';

else

for (j=0;j<keylen;j++)

key[j]=key1[j];

for (j=keylen-1;j>0;j--) {

if(temp[j]==key[j])

rem[j-1]='0'; else

rem[j-1]='1';

}

rem[keylen-1]=input[i+keylen];

strcpy(temp,rem);

}

strcpy(rem,temp);

printf("\nQuotient is ");

for (i=0;i<msglen;i++)

printf("%c",quot[i]);

printf("\nRemainder is ");

for (i=0;i<keylen-1;i++)

printf("%c",rem[i]);

printf("\nFinal data is: ");

for (i=0;i<msglen;i++)

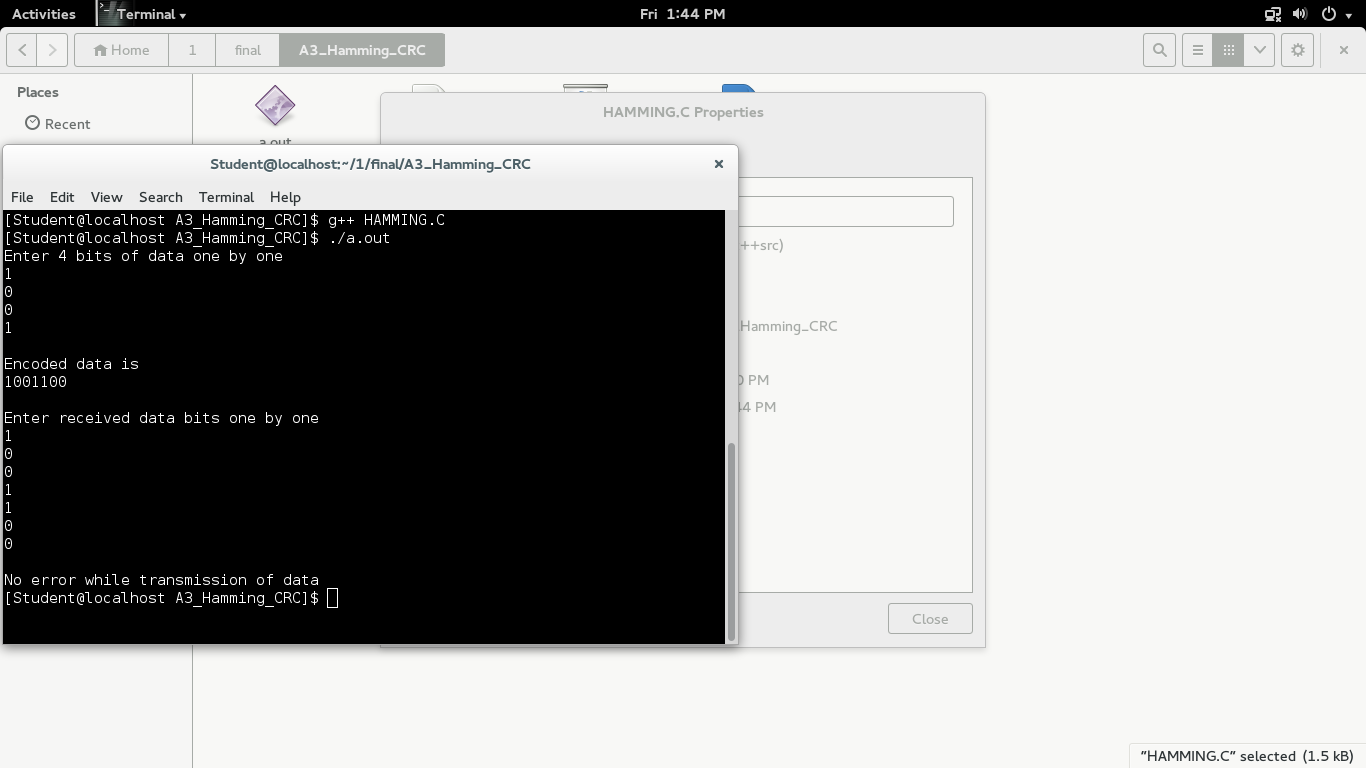
printf("%c",input[i]);

for (i=0;i<keylen-1;i++)

printf("%c",rem[i]);

}

\*\*\* OUTPUT of CRC \*\*\*



/\*\*\*\*\*\* A3 Hamming \*\*\*\*\*\*/

#include<stdio.h>

int main()

{

int data[10];

int dataatrec[10],c,c1,c2,c3,i;

printf("Enter 4 bits of data one by one\n");

scanf("%d",&data[0]);

scanf("%d",&data[1]);

scanf("%d",&data[2]);

scanf("%d",&data[4]);

//Calculation of even parity

data[6]=data[0]^data[2]^data[4];

data[5]=data[0]^data[1]^data[4];

data[3]=data[0]^data[1]^data[2];

printf("\nEncoded data is\n");

for(i=0;i<7;i++)

printf("%d",data[i]);

printf("\n\nEnter received data bits one by one\n");

for(i=0;i<7;i++)

scanf("%d",&dataatrec[i]);

c1=dataatrec[6]^dataatrec[4]^dataatrec[2]^dataatrec[0];

c2=dataatrec[5]^dataatrec[4]^dataatrec[1]^dataatrec[0];

c3=dataatrec[3]^dataatrec[2]^dataatrec[1]^dataatrec[0];

c=c3\*4+c2\*2+c1 ;

if(c==0)

{

printf("\nNo error while transmission of data\n");

}

else

{

printf("\nError on position %d",c);

printf("\nData sent : ");

for(i=0;i<7;i++)

printf("%d",data[i]);

printf("\nData received : ");

for(i=0;i<7;i++)

printf("%d",dataatrec[i]);

printf("\nCorrect message is\n");

//if errorneous bit is 0 we complement it else vice versa

if(dataatrec[7-c]==0)

dataatrec[7-c]=1;

else

dataatrec[7-c]=0;

for (i=0;i<7;i++)

{

printf("%d",dataatrec[i]);

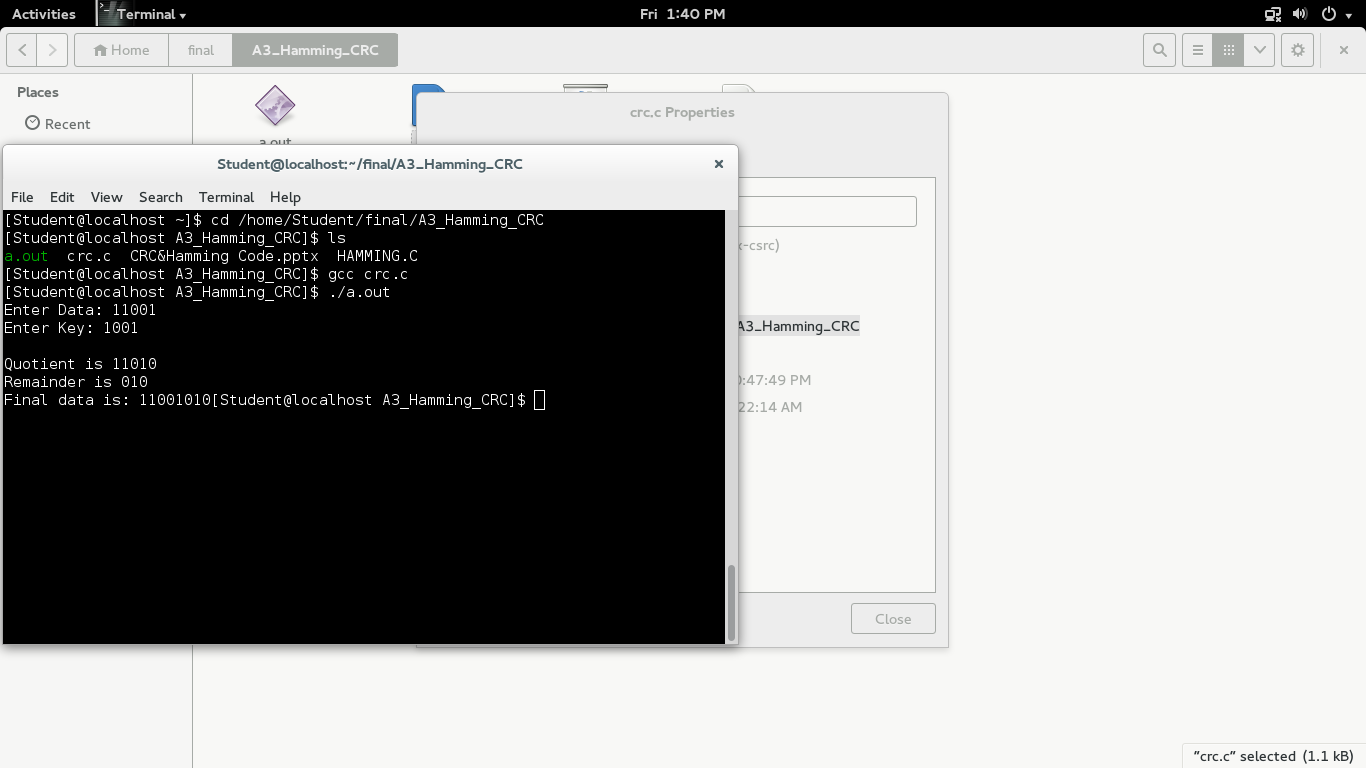
}

}

return 0;

}

\*\*\* OUTPUT \*\*\*



/\*\*\*\*\* A4.selective repeat \*\*\*\*\*/

//Client

import java.lang.System;

import java.net.\*;

import java.io.\*;

public class Client {

static Socket connection;

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

try {

int v[] = new int[8];

//int g[] = new int[8];

int n = 0;

InetAddressaddr = InetAddress.getByName("Localhost");

System.out.println(addr);

connection = new Socket(addr, 8011);

DataOutputStream out = new DataOutputStream(

connection.getOutputStream());

DataInputStream in = new DataInputStream(

connection.getInputStream());

int p = in.read();

System.out.println("No of frame is:" + p);

for (int i = 0; i< p; i++) {

v[i] = in.read();

System.out.println(v[i]);

//g[i] = v[i];

}

v[5] = -1;

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

{

System.out.println("Received frame is: " + v[i]);

}

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

if (v[i] == -1) {

System.out.println("Request to retransmit from packet no "

+ (i+1) + " again!!");

n = i;

out.write(n);

out.flush();

}

System.out.println();

v[n] = in.read();

System.out.println("Received frame is: " + v[n]);

System.out.println("quiting");

} catch (Exception e) {

System.out.println(e);

}

}

}

//Server

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.SocketException;

public class Server {

static ServerSocketServersocket;

static DataInputStream dis;

static DataOutputStream dos;

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

try {

int a[] = { 30, 40, 50, 60, 70, 80, 90, 100 };

Serversocket = new ServerSocket(8011);

System.out.println("waiting for connection");

Socket client = Serversocket.accept();

dis = new DataInputStream(client.getInputStream());

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

System.out.println("The number of packets sent is:" + a.length);

int y = a.length;

dos.write(y);

dos.flush();

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

dos.write(a[i]);

dos.flush();

}

int k = dis.read();

dos.write(a[k]);

dos.flush();

} catch (IOException e) {

System.out.println(e);

} finally {

try {

dis.close();

dos.close();

} catch (IOException e) {

// TODO Auto-generated catch block

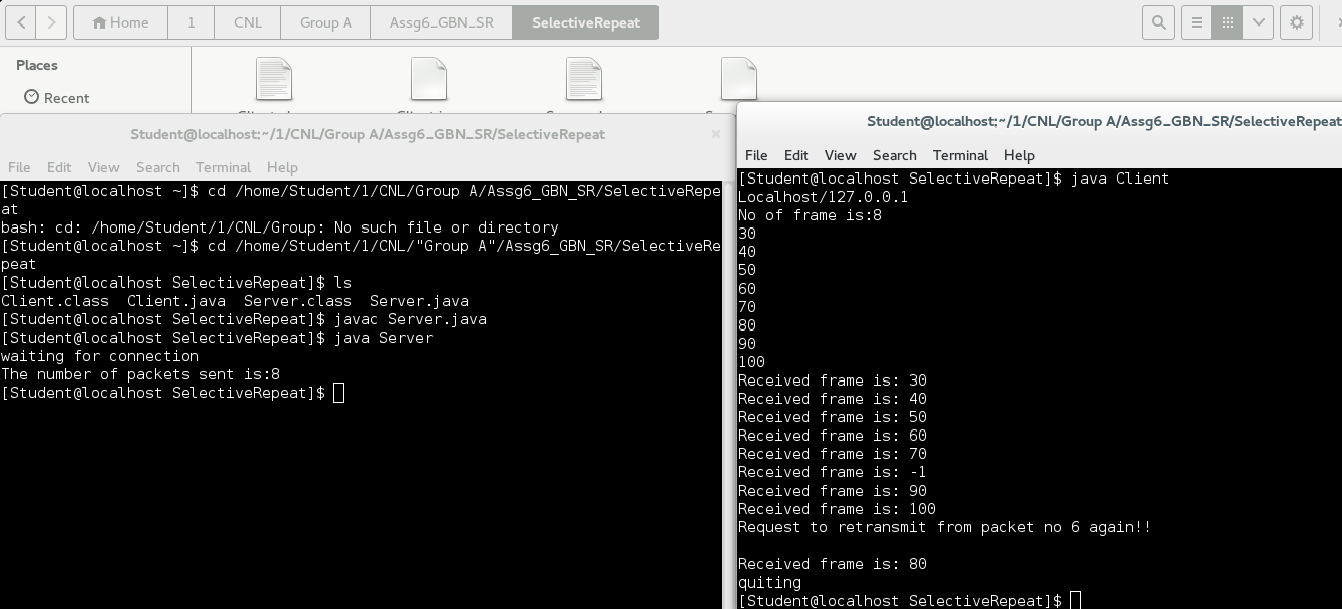
e.printStackTrace();

}

}

}

}



/\*\*\*\*\* A4.sliding window go back N\*\*\*\*\*/

//Client

import java.io.\*;

import java.net.\*;

import java.math.\*;

import java.util.\*;

class testclient

{

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

{

InetAddressaddr=InetAddress.getByName("Localhost");

System.out.println(addr);

Socket connection=new Socket(addr,5000);

BufferedInputStream in=new BufferedInputStream(connection.getInputStream());

DataOutputStream out=new DataOutputStream(connection.getOutputStream());

Scanner scr=new Scanner(System.in);// this will be used to accept i/p from console

System.out.println(" client is Connected to server" + addr);

System.out.println("Enter the number of frames to be requested to the server");

int c=scr.nextInt();

out.write(c); // write no of frames on client socket

out.flush();

System.out.println("Enter the type of trans. Error=1 ; No Error=0");

int choice=scr.nextInt();

out.write(choice); //write choice on socket

int check=0;

int i=0;

int j=0;

if(choice==0)

{

for(j=0;j<c;++j)

{

i=in.read(); //read all frames one by one from server

System.out.println("received frame no: "+i);

System.out.println("Sending acknowledgement for frame no: "+i);

out.write(i); //write ack to socket

out.flush();

}

out.flush();

}

else

{

for(j=0;j<c;++j)

{

i=in.read(); //read 0,1,2,3 frame

if(i==check)

{

System.out.println("received frame no: "+i);

System.out.println("Sending acknowledgement for frame no: "+i);

out.write(i); //sent ack of frame 0,1

++check;

}

else

{

--j;

System.out.println("Discarded frame no: "+i);

System.out.println("Sending NEGATIVE ack");

out.write(-1);

}

out.flush();

}

}//end of else for error

in.close();

out.close();

System.out.println("Quiting");

}// end of main method

}// end of main class

/\*\*\*\*\* A4.sliding window go back N \*\*\*\*\*/

//server

import java.io.\*;

import java.net.\*;

import java.util.\*;

class testserver

{

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

{

System.out.println("server Waiting for connection....");

InetAddressaddr=InetAddress.getByName("Localhost");

ServerSocket ss=new ServerSocket(5000);

Socket client=new Socket();

client=ss.accept();

BufferedInputStream in=new BufferedInputStream(client.getInputStream());

DataOutputStream out=new DataOutputStream(client.getOutputStream());

System.out.println("Received request for sending frames");

int p=in.read(); //read no of frames sent by client

booleanf[]=new boolean[p];

int pc=in.read(); //read choice sent by client

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

if(pc==0)

{

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

{

System.out.println("sending frame number "+i);

out.write(i); //send frame on server socket

out.flush();

System.out.println("Waiting for acknowledgement");

try

{

Thread.sleep(7000);

}

catch(Exception e){}

int a=in.read(); //read ack on servers socket from client

System.out.println("received acknowledgement for frame "+i+" as "+a);

}

out.flush();

}

else

{

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

{

if(i==2)

{

System.out.println("sending frame no "+i); //sent frame 2

}

else

{

System.out.println("sending frame no "+i);

out.write(i); //write 0 and 1 and 3 frame

out.flush();

System.out.println("Waiting for acknowledgement ");

try

{

Thread.sleep(7000);

}

catch(Exception e){}

int a=in.read(); //Read NACK

if(a!=255)

{

System.out.println("received ack for frame no: "+i+" as "+a);

f[i]=true;

}

}// end of inner else

}// end of for

// check which frames have not been ack

for(int a=0;a<p;++a)

{

if(f[a]==false)

{

System.out.println("Resending frame "+a);

out.write(a);

out.flush();

System.out.println("Waiting for ack ");

try

{

Thread.sleep(5000);

}

catch(Exception e){}

int b=in.read();

System.out.println("received ack for frame no: "+a+" as "+b);

f[a]=true;

}

}

out.flush();

}// end of else which is for error

in.close();

out.close();

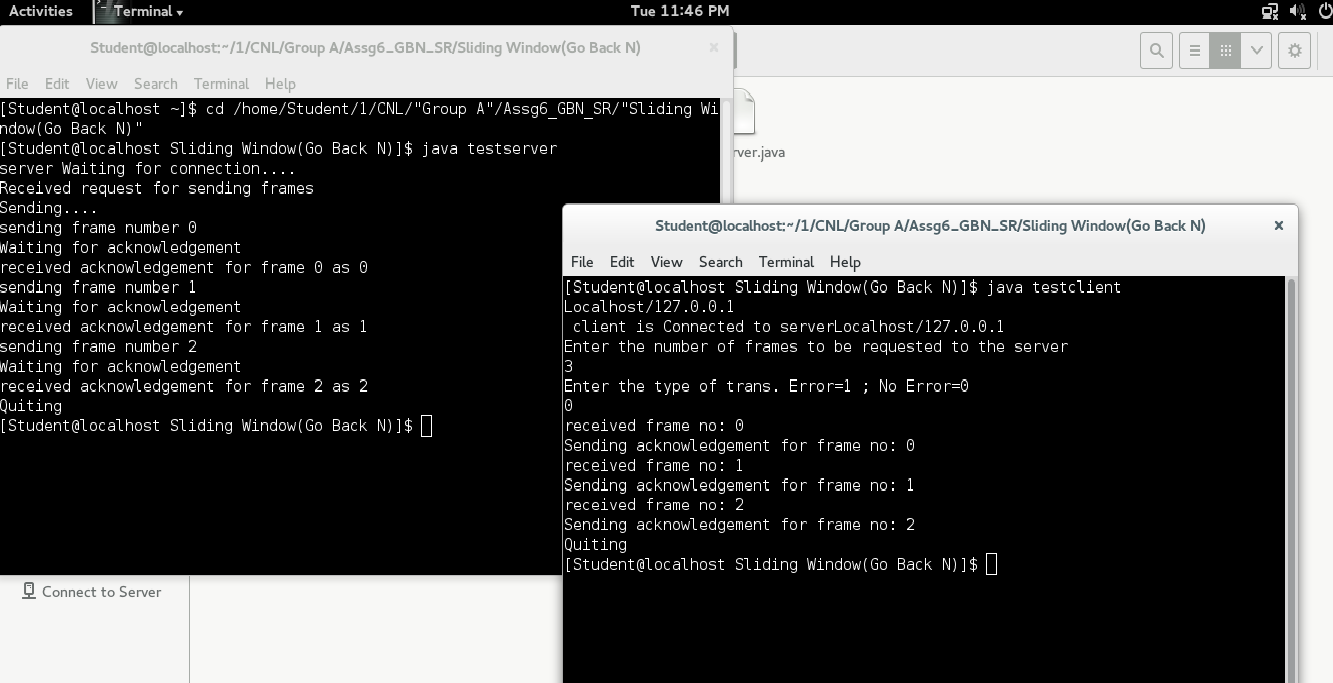
client.close();

ss.close();

System.out.println("Quiting");

}// end main method

}// end main class



/\*\*\*\*\* A5.Subneting \*\*\*\*\*/

import java.util.Scanner;

class Subnet{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter the ip address: ");

String ip = sc.nextLine();

String split\_ip[] = ip.split("\\."); //SPlit the string after every .

String split\_bip[] = new String[4]; //split binary ip

String bip = "";

String mask1="";

int firstoctet = Integer.parseInt(split\_ip[0]);

if(firstoctet<=127)

{

mask1 = "255.0.0.0";

System.out.println("Class A IP Address");

System.out.println("Default mask is: "+mask1);

}

else if(firstoctet>=128 &&firstoctet<=191)

{

mask1 = "255.255.0.0";

System.out.println("Class B IP Address");

System.out.println("Default mask is: "+mask1);

}

else if(firstoctet>=192 &&firstoctet<=223)

{

mask1 = "255.255.255.0";

System.out.println("Class C IP Address");

System.out.println("Defaultt mask is: "+mask1);

}

for(int i=0;i<4;i++)

{

split\_bip[i] = appendZeros(Integer.toBinaryString(Integer.parseInt(split\_ip[i]))); // "18" => 18 => 10010 => 00010010

bip += split\_bip[i];

}

System.out.println("IP in binary is "+bip);

System.out.print("Enter the number of subnets: ");

int n = sc.nextInt();

//Calculation of mask

double bits = (int)Math.ceil(Math.log(n)/Math.log(2)); /\*eg if address = 120, log 120/log 2 gives log to the base 2 => 6.9068, ceil gives us upper integer \*/

int y=(int) bits;

System.out.println("Number of bits borrowd from host to network are = "+y);

int z=8-y;

System.out.println("Number of host bits are = "+z);

//int mask = 32-bits;

double mask=256-(Math.pow(2.0,z));

int x = (int) mask;

System.out.println("New subnet mask is = "+x);

int sub\_size=256-x;

System.out.println("Subnet Size = "+sub\_size);

//Calculation of first address and last address

int fbip[] = new int[32];

for(int i=0; i<32;i++)

fbip[i] = (int)bip.charAt(i)-48; //convert cahracter 0,1 to integer 0,1

for(int i=31;i>31-z;i--)//Get first address by ANDing last n bits with 0

fbip[i] &= 0;

String fip[] ={"","","",""};

for(int i=0;i<32;i++)

fip[i/8] = new String(fip[i/8]+fbip[i]);

System.out.println("first Subnet Details");

System.out.print("first N/W address is = ");

for(int i=0;i<4;i++){

System.out.print(Integer.parseInt(fip[i],2));

if(i!=3) System.out.print(".");

}

System.out.println();

int lbip[] = new int[32];

for(int i=0; i<32;i++)

lbip[i] = (int)bip.charAt(i)-48; //convert cahracter 0,1 to integer 0,1

for(int i=31;i>31-z;i--)//Get last address by ORing last n bits with 1

lbip[i] |= 1;

String lip[] = {"","","",""};

for(int i=0;i<32;i++)

lip[i/8] = new String(lip[i/8]+lbip[i]);

System.out.print("Broadcast address is = ");

for(int i=0;i<4;i++){

System.out.print(Integer.parseInt(lip[i],2));

if(i!=3) System.out.print(".");

}

System.out.println();

}

static String appendZeros(String s)

{

String temp = new String("00000000");

return (temp.substring(s.length())+ s);

}

}

\*\*\*OUTPUT\*\*\*

[Student@localhost Assg7\_Subnetting]$ java Subnet

Enter the ip address: 192.168.1.1

Class C IP Address

Defaultt mask is: 255.255.255.0

IP in binary is 11000000101010000000000100000001

Enter the number of subnets: 255.255.255.0

ABRT problem creation: 'success'

Exception in thread "main" java.util.InputMismatchException

at java.util.Scanner.throwFor(Scanner.java:909)

at java.util.Scanner.next(Scanner.java:1530)

at java.util.Scanner.nextInt(Scanner.java:2160)

at java.util.Scanner.nextInt(Scanner.java:2119)

at Subnet.main(Subnet.java:39)

[Student@localhost Assg7\_Subnetting]$ java Subnet

Enter the ip address: 172.16.0.51

Class B IP Address

Default mask is: 255.255.0.0

IP in binary is 10101100000100000000000000110011

Enter the number of subnets: 2

Number of bits borrowd from host to network are = 1

Number of host bits are = 7

New subnet mask is = 128

Subnet Size = 128

first Subnet Details

first N/W address is = 172.16.0.0

Broadcast address is = 172.16.0.127

[Student@localhost Assg7\_Subnetting]$

A7

Tcp client server

Server : -

package cnmock;

import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.InputStreamReader;

import java.net.ServerSocket;

import java.net.Socket;

public class ServerChat

{

public static void main(String [] args)

{

try

{

System.out.println("Server");

ServerSocket ss = new ServerSocket(4444);

Socket client = ss.accept();

DataInputStream dis = new DataInputStream(client.getInputStream());

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

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

String clmsg = dis.readUTF();

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

System.out.println("Enter the message : ");

String msg = br.readLine();

dos.writeUTF(msg);

client.close();

ss.close();

}

catch (Exception e) {

e.printStackTrace();

}

}

Client : -

package cnmock;

import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.InputStreamReader;

import java.net.Socket;

public class ClientChat {

public static void main(String [] args)

{

try

{

System.out.println("Client");

//ServerSocket ss = new ServerSocket(4444);

Socket client = new Socket("localhost",4444);

DataInputStream dis = new DataInputStream(client.getInputStream());

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

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

System.out.println("Enter the message : ");

String srmsg = br.readLine();

dos.writeUTF(srmsg);

System.out.println("Server : " + dis.readUTF());

client.close();

}

catch (Exception e)

{

e.printStackTrace();

}

}

}

Tcp calculator

Server : -

package cnmock;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.FileInputStream;

import java.io.OutputStream;

import java.net.ServerSocket;

import java.net.Socket;

public class ServerCal

{

public static void main(String[] args)

{

try

{

System.out.println("Server");

ServerSocket ss = new ServerSocket(4444);

Socket client = ss.accept();

DataInputStream dis = new DataInputStream(client.getInputStream());

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

StringBuffer buffer1 = new StringBuffer();

buffer1.append("Menu");

buffer1.append("\n1.Add");

buffer1.append("\n2.Subtract");

buffer1.append("\n3.Multiply");

buffer1.append("\n4.Divide");

int ch ;

dos.writeUTF("start");

dos.writeUTF(buffer1.toString());

ch = dis.readInt();

System.out.println("Choice : " + ch);

switch (ch) {

case 1:

{

StringBuffer buffer = new StringBuffer();

buffer.append("\nAddition");

buffer.append("\nEnter num1 and num2 for addition : ");

dos.writeUTF(buffer.toString());

int x,y,z;

x = dis.readInt();

y = dis.readInt();

z = x + y;

dos.writeInt(z);

}

break;

case 2:

{

StringBuffer buffer = new StringBuffer();

buffer.append("\nSubtraction");

buffer.append("\nEnter num1 and num2 for subtraction : ");

dos.writeUTF(buffer.toString());

int x,y,z;

x = dis.readInt();

y = dis.readInt();

z = x - y;

dos.writeInt(z);

}

break;

case 3:

{

StringBuffer buffer = new StringBuffer();

buffer.append("\nMultiplition");

buffer.append("\nEnter num1 and num2 for multiplition : ");

dos.writeUTF(buffer.toString());

int x,y,z;

x = dis.readInt();

y = dis.readInt();

z = x \* y;

dos.writeInt(z);

}

break;

case 4:

{

StringBuffer buffer = new StringBuffer();

buffer.append("\nDivision");

buffer.append("\nEnter num1 and num2 for division : ");

dos.writeUTF(buffer.toString());

int x,y,z;

x = dis.readInt();

y = dis.readInt();

if(x!=0 && y!=0)

{

z = x / y;

dos.writeInt(z);

}

else

if(x==0 || y==0)

{

dos.writeUTF("Divide zero error");

}

else

if(x<0 || y<0)

{

dos.writeUTF("no negetive integer allowed for division");

}

}

break;

default:

{

dos.writeUTF("bye");

System.out.println("Select fom the menu only");

}

break;

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

Client : -

package cnmock;

import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.net.Socket;

public class ClientCal

{

public static void main(String[] args)

{

try

{

System.out.println("Client");

//ServerSocket ss = new ServerSocket(4444);

Socket client = new Socket("localhost",4444);

DataInputStream dis = new DataInputStream(client.getInputStream());

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

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

String start = dis.readUTF();

String serverMessage = dis.readUTF();

System.out.println(serverMessage + "\nEnter your choice : ");

String ch = br.readLine();

dos.writeInt(Integer.parseInt(ch));

serverMessage = dis.readUTF();

System.out.println(serverMessage);

String x,y;

x = br.readLine();

y=br.readLine();

dos.writeInt(Integer.parseInt(x));

dos.writeInt(Integer.parseInt(y));

System.out.println("Result : " + dis.readInt());

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

Tcp File Transfer : -

Server :

**package**cnmock;

**import**java.io.FileInputStream;

**import**java.io.OutputStream;

**import**java.net.ServerSocket;

**import**java.net.Socket;

**publicclass**ServerFile

{

**publicstaticvoid**main(String[] args)

{

**try**

{

System.out.println("Server");

ServerSocketss = **new**ServerSocket(4444);

Socket client = ss.accept();

FileInputStreamfis = **new**FileInputStream("D:\\zzsample\\inputfile.txt");

**byte** [] bt = **newbyte**[2000];

fis.read(bt,0,bt.length);

**OutputStream**ois = client.getOutputStream();

ois.write(bt, 0, bt.length);

System.out.println("Byte : " + bt);

}

**catch**(Exception e)

{

e.printStackTrace();

}

}

}

Client :

**package**cnmock;

**import**java.io.FileInputStream;

**import**java.io.FileOutputStream;

**import**java.io.InputStream;

**import**java.io.OutputStream;

**import**java.net.ServerSocket;

**import**java.net.Socket;

**publicclass**ClientFile

{

**publicstaticvoid**main(String [] main)

{

**try**

{

System.out.println("Client");

//ServerSocketss = new ServerSocket(4444);

Socket client = **new** Socket("localhost",4444);

FileOutputStreamfis = **new**FileOutputStream("D:\\zzsample\\outputfile.txt");

**InputStream**iis = client.getInputStream();

**byte** [] bt = **newbyte**[2000];

iis.read(bt, 0, bt.length);

System.out.println("Byte : " + bt);

fis.write(bt, 0, bt.length);

}

**catch**(Exception e)

{

e.printStackTrace();

}

}

}

/\*\*\* A8. UDP Socket client \*\*\*/

import java.io.BufferedOutputStream;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.net.InetAddress;

import java.net.Socket;

public class FileTransferClient {

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.

FileOutputStreamfos = new FileOutputStream("/home/Student/1/final/A8\_udp\_socket/file2.txt");

BufferedOutputStreambos = 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!");

}

}

/\*\*\* A8. UDP socket 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 FileTransferServer {

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

//Initialize Sockets

ServerSocketssock = new ServerSocket(5000);

Socket socket = ssock.accept();

//The InetAddress specification

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

//Specify the file

File file = new File("/home/Student/1/final/A8\_udp\_socket/file.txt");

FileInputStreamfis = new FileInputStream(file);

BufferedInputStream bis = new BufferedInputStream(fis);

//Get socket's output stream

OutputStreamos = 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!");

}

}

\*\*\* OUTPUT\*\*\*



/\*\*\* A9. Packet Format \*\*\*/

#include <iostream>

#include <pcap.h>

#include <net/ethernet.h>

#include <netinet/ip.h>

#include <netinet/in.h>

#include <netinet/tcp.h>

#include <arpa/inet.h>

using namespace std;

void packetHandler(u\_char \*userData, const struct pcap\_pkthdr\* pkthdr, const u\_char\* packet);

int main() {

pcap\_t \*descr;

char errbuf[PCAP\_ERRBUF\_SIZE];

// open capture file for offline processing

descr = pcap\_open\_offline("http.cap", errbuf);

if (descr == NULL) {

cout<< "pcap\_open\_live() failed: " <<errbuf<<endl;

return 1;

}

cout<<"Source Ip\t\t"<<"Source Port\t\t"<<"Destination Ip\t\t"<<"Destination port\t\n";

// start packet processing loop, just like live capture

if (pcap\_loop(descr, 0, packetHandler, NULL) < 0) {

cout<< "pcap\_loop() failed: " <<pcap\_geterr(descr);

return 1;

}

cout<< "capture finished" <<endl;

return 0;

}

void packetHandler(u\_char \*userData, const struct pcap\_pkthdr\* pkthdr, const u\_char\* packet) {

const struct ether\_header\* ethernetHeader;

const struct ip\* ipHeader;

const struct tcphdr\* tcpHeader;

char sourceIp[INET\_ADDRSTRLEN];

char destIp[INET\_ADDRSTRLEN];

u\_intsourcePort, destPort;

u\_char \*data;

int dataLength = 0,ipflag=0;

string dataStr = "";

ethernetHeader = (struct ether\_header\*)packet;

if (ntohs(ethernetHeader->ether\_type) == ETHERTYPE\_IP) {

ipflag=1;

ipHeader = (struct ip\*)(packet + sizeof(struct ether\_header));

inet\_ntop(AF\_INET, &(ipHeader->ip\_src), sourceIp, INET\_ADDRSTRLEN);

inet\_ntop(AF\_INET, &(ipHeader->ip\_dst), destIp, INET\_ADDRSTRLEN);

if (ipHeader->ip\_p == IPPROTO\_TCP) {

tcpHeader = (tcphdr\*)(packet + sizeof(struct ether\_header) + sizeof(struct ip));

sourcePort = ntohs(tcpHeader->source);

destPort = ntohs(tcpHeader->dest);

data = (u\_char\*)(packet + sizeof(struct ether\_header) + sizeof(struct ip) + sizeof(struct tcphdr));

dataLength = pkthdr->len - (sizeof(struct ether\_header) + sizeof(struct ip) + sizeof(struct tcphdr));

// convert non-printable characters, other than carriage return, line feed,

// or tab into periods when displayed.

for (int i = 0; i<dataLength; i++) {

if ((data[i] >= 32 && data[i] <= 126) || data[i] == 10 || data[i] == 11 || data[i] == 13) {

dataStr += (char)data[i];

} else {

dataStr += ".";

}

}

// print the results

cout<<sourceIp<< "\t\t"<<sourcePort<< "\t\t" <<destIp<<"\t\t" <<destPort<<"\n";

if(ipflag==1)

{

cout<<"protocols : IP and TCP";

}

if (dataLength> 0) {

// cout<<dataStr<<endl;

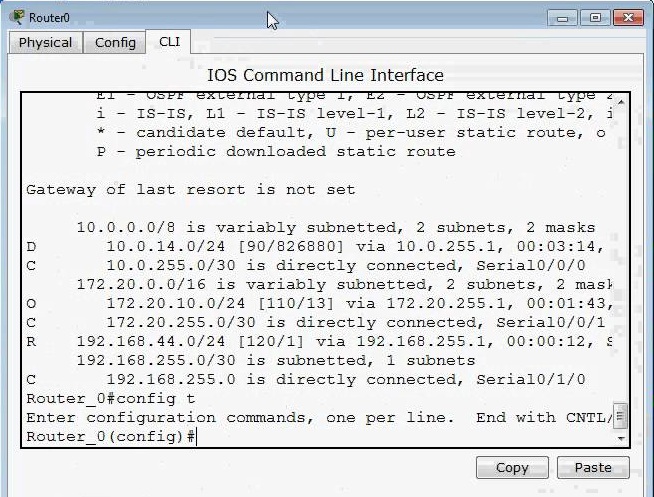
}

}

}

}

\*\*\* OUTPUT of Packet format \*\*\*



/\*\*\* A11. DNS \*\*\*/

import socket

print 'Welcome to DNS Lookup'

print 'Enter your option 1. URL to IP 2. IP to URL'

op=raw\_input('Enter Option')

var=raw\_input('Enter URL/IP')

addr1 = socket.gethostbyname(var)

addr6=socket.gethostbyaddr(var)

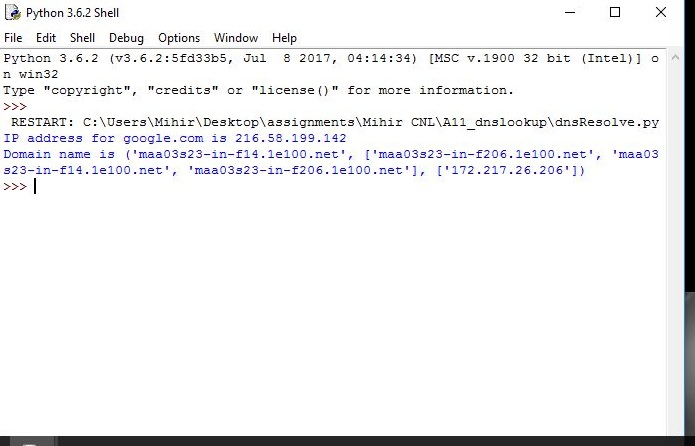
if op==1:

print(addr1)

else:

print(addr6)

\*\*\* OUTPUT \*\*\*



/\*\*\* B2. NS2 for

#Create a simulator object

set ns [new Simulator]

#Define different colors for data flows

$ns color 1 Blue

$ns color 2 Red

#Open the nam trace file

set nf [open out.nam w]

$ns namtrace-all $nf

#Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

#Close the trace file

close $nf

#Execute nam on the trace file

exec namout.nam&

exit 0

}

#Create four nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

#Create links between the nodes

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

$ns duplex-link $n1 $n2 1Mb 10ms DropTail

#Create a TCP agent and attach it to node n0

set tcp0 [new Agent/TCP]

$tcp0 set class\_ 1

$ns attach-agent $n0 $tcp0

# Create a CBR traffic source and attach it to tcp0

set ftp [new Application/FTP]

$ftp attach-agent $tcp0

#Create a Null agent (a traffic sink) and attach it to node n3

set null0 [new Agent/TCPSink]

$ns attach-agent $n2 $null0

#Connect the traffic sources with the traffic sink

$ns connect $tcp0 $null0

#Schedule events for the CBR agents

$ns at 0.5 "$ftp start"

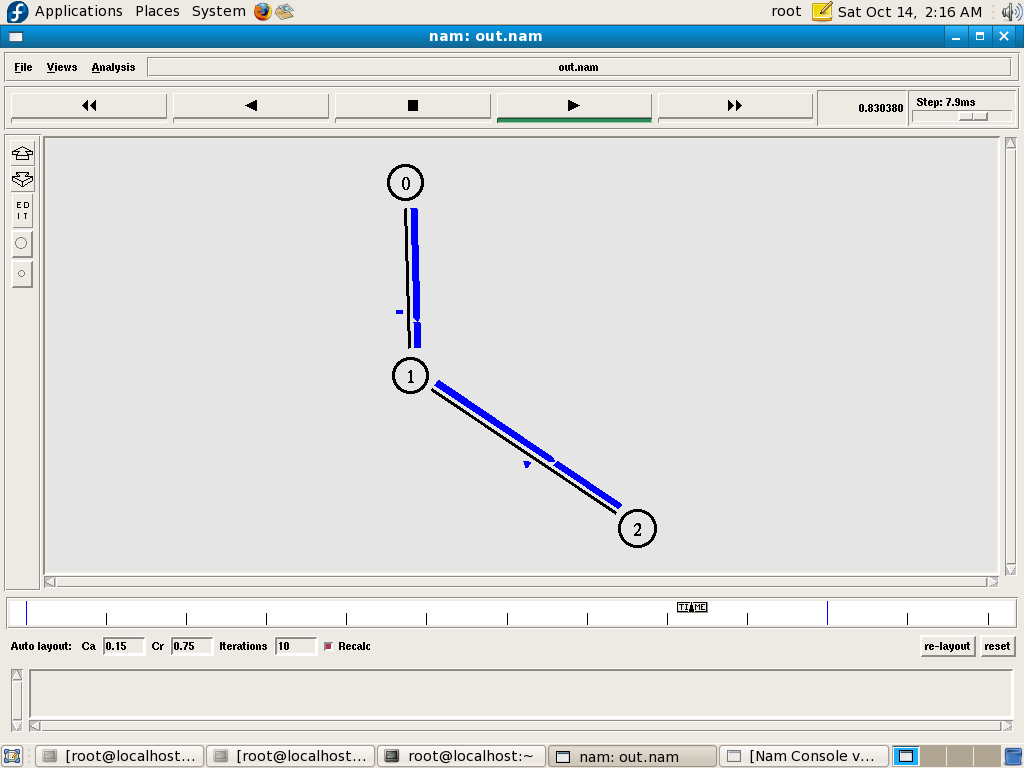
$ns at 1.0 "$ftp stop"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns run



/\*\*\* B3.A.Multiuser chat client \*\*\*/

import java.io.\*;

import java.net.\*;

public class multclient

{

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

String sentence;

String modifiedSentence;

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

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

while (true)

{

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

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

sentence = inFromUser.readLine();

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

if (sentence.equals("EXIT")) {

break;

}

modifiedSentence = inFromServer.readLine();

System.out.println("FROM SERVER: " + modifiedSentence);

}

clientSocket.close();

}

}

/\*\*\* B3.A.Multiuser chat server\*\*\*/

import java.io.\*;

import java.net.\*;

public class multserver

{

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

{

ServerSocketwelcomeSocket = new ServerSocket(6789);

Responder h = new Responder();

// server runs for infinite time and

// waits for clients to connect

while (true)

{

Socket connectionSocket = welcomeSocket.accept();

// on connection establishment start a new thread for each client

// each thread shares a common responder object

// which will be used to respond every client request

Thread t = new Thread(new MyServer(h, connectionSocket));

// start thread

t.start();

}

}

}

class MyServer implements Runnable

{

Responder h;

Socket connectionSocket;

public MyServer(Responder h, Socket connectionSocket)

{

this.h = h;

this.connectionSocket = connectionSocket;

}

//@Override

public void run()

{

while (h.responderMethod(connectionSocket))

{

try

{

// once an conversation with one client done,

// give chance to other threads

// so make this thread sleep

Thread.sleep(5000);

} catch (InterruptedException ex) { }

}

try {

connectionSocket.close();

} catch (IOException ex) { }

}

}

class Responder

{

String serverSentence;

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

// on client process termination or

// client sends EXIT then to return false to close connection

// else return true to keep connection alive

// and continue conversation

synchronized public booleanresponderMethod(Socket connectionSocket)

{

try {

BufferedReaderinFromClient =new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));

DataOutputStreamoutToClient =new DataOutputStream(connectionSocket.getOutputStream());

String clientSentence = inFromClient.readLine();

// if client process terminates it get null, so close connection

if (clientSentence == null || clientSentence.equals("EXIT")) {

return false;

}

if (clientSentence != null) {

System.out.println("client : " + clientSentence);

}

serverSentence = br.readLine() + "\n";

outToClient.writeBytes(serverSentence);

return true;

} catch (SocketException e) {

System.out.println("Disconnected");

return false;

} catch (Exception e) {

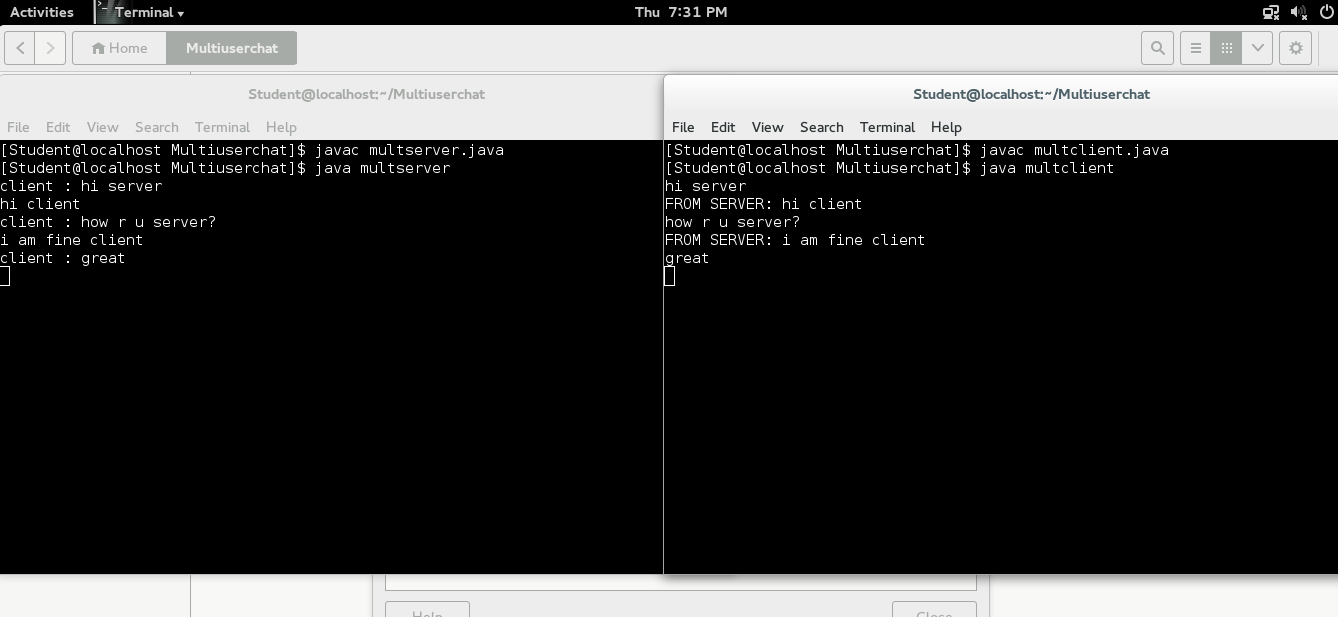
e.printStackTrace();

return false;}

}

}

\*\*\* OUTPUT of Multiuser chat \*\*\*



/\*\*\* B3.B.Peer to peer chat client\*\*\*/

import java.io.\*;

import java.net.\*;

import java.net.Socket;

class client

{

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

{

String sentence,s1;

String modifiedSentence;

BufferedReaderinFromUser =

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

Socket clientSocket = new Socket("127.0.0.1", 6666);

DataOutputStreamoutToServer =

new DataOutputStream(clientSocket.getOutputStream());

BufferedReaderinFromServer =

new BufferedReader(new

InputStreamReader(clientSocket.getInputStream()));

System.out.println("Enter your Message:");

sentence = inFromUser.readLine();

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

s1 = inFromServer.readLine();

System.out.println("FROM SERVER: " + s1);

clientSocket.close();

}

}

/\*\*\* B3.B.Peer to peer chat server\*\*\*/

import java.io.\*;

import java.net.\*;

import java.net.ServerSocket;

class server

{

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

{

String clientSentence;

String sentence;

ServerSocketwelcomeSocket = new ServerSocket(6666);

Socket connectionSocket = welcomeSocket.accept();

BufferedReaderinFromClient =

new BufferedReader(new

InputStreamReader(connectionSocket.getInputStream()));

DataOutputStreamoutToClient =

new DataOutputStream(connectionSocket.getOutputStream());

BufferedReaderinFromUser =

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

clientSentence = inFromClient.readLine();

System.out.println("FROM CLIENT: " +clientSentence );

System.out.println("Enter your Message:");

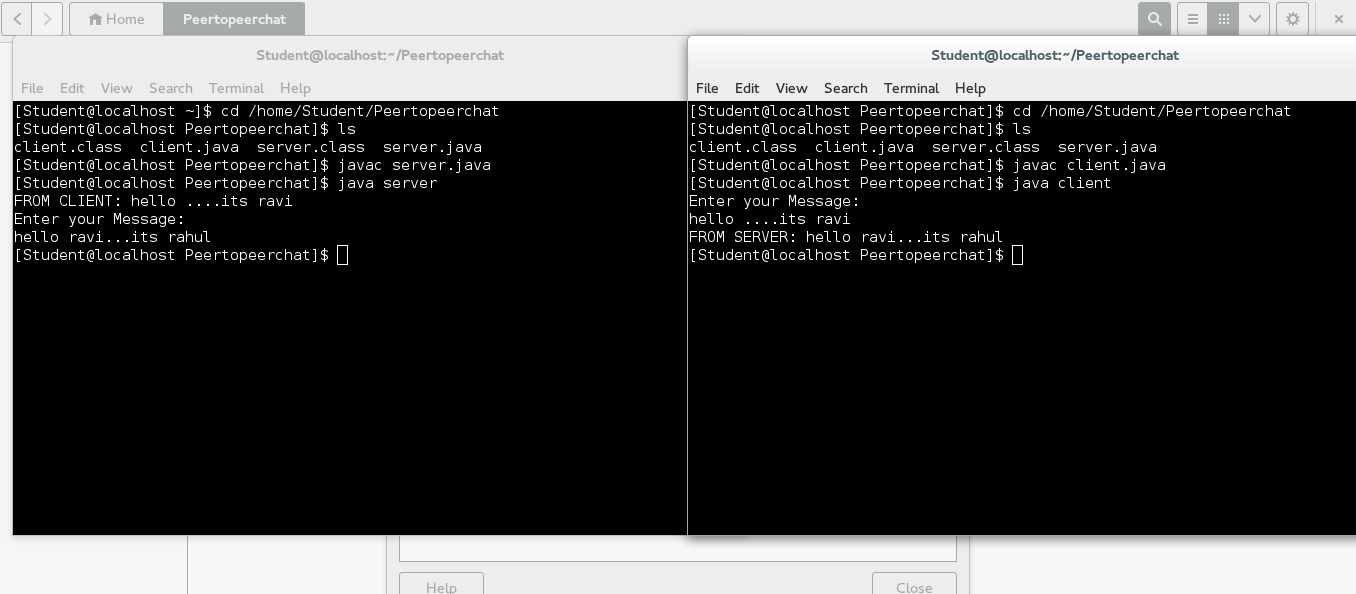
sentence = inFromUser.readLine();

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

}

}

\*\*\* OUTPUT of B3.B \*\*\*



/\*\*\* B4.A. UDp chat multiuser client \*\*\*/

import java.net.\*;

import java.io.\*;

public class Client {

public static void main(String[] args) {

Socket clientSocket = null;

DataInputStream is = null;

PrintStreamos = null;

DataInputStreaminputLine = null;

/\*

\* Open a socket on port 2222. Open the input and the output streams.

\*/

try {

clientSocket = new Socket("localhost", 2222);

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

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

inputLine = new DataInputStream(new BufferedInputStream(System.in));

} catch (UnknownHostException e) {

System.err.println("Don't know about host");

} catch (IOException e) {

System.err.println("Couldn't get I/O for the connection to host");

}

/\*

\* If everything has been initialized then we want to write some data to the

\* socket we have opened a connection to on port 2222.

\*/

if (clientSocket != null &&os != null && is != null) {

try {

/\*

\* Keep on reading from/to the socket till we receive the "Ok" from the

\* server, once we received that then we break.

\*/

System.out.println("The client started. Type any text. To quit it type 'Ok'.");

String responseLine;

os.println(inputLine.readLine());

while ((responseLine = is.readLine()) != null) {

System.out.println(responseLine);

if (responseLine.indexOf("Ok") != -1) {

break;

}

os.println(inputLine.readLine());

}

/\*

\* Close the output stream, close the input stream, close the socket.

\*/

os.close();

is.close();

clientSocket.close();

} catch (UnknownHostException e) {

System.err.println("Trying to connect to unknown host: " + e);

} catch (IOException e) {

System.err.println("IOException: " + e);

}

}

}

}

/\*\*\* B4.A. UDP chat multiuser server \*\*\*/

import java.io.\*;

import java.net.\*;

public class Server {

public static void main(String args[]) {

ServerSocketechoServer = null;

String line;

DataInputStream is;

PrintStreamos;

Socket clientSocket = null;

/\*

\* Open a server socket on port 2222. Note that we can't choose a port less

\* than 1023 if we are not privileged users (root).

\*/

try {

echoServer = new ServerSocket(2222);

} catch (IOException e) {

System.out.println(e);

}

/\*

\* Create a socket object from the ServerSocket to listen to and accept

\* connections. Open input and output streams.

\*/

System.out.println("The server started. To stop it press <CTRL><C>.");

try {

clientSocket = echoServer.accept();

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

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

/\* As long as we receive data, echo that data back to the client. \*/

while (true) {

line = is.readLine();

os.println("From server: " + line);

}

} catch (IOException e) {

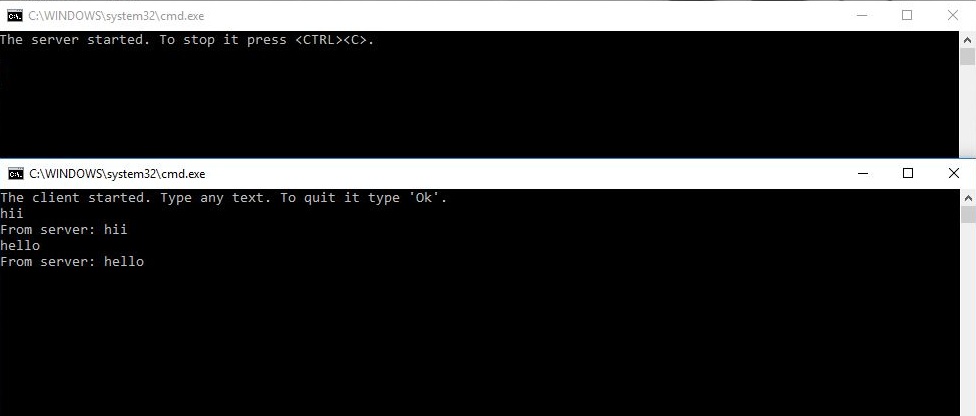
System.out.println(e);

}

}

}

\*\*\* OUTPUT of B4.A.Multiuser chat \*\*\*



/\*\*\* B4.B. UDP chat p2p client \*\*\*/

import java.io.\*;

import java.net.\*;

class UDPClient

{

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

{

BufferedReaderinFromUser =

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

DatagramSocketclientSocket = new DatagramSocket();

InetAddressIPAddress = InetAddress.getByName("localhost");

byte[] sendData = new byte[1024];

byte[] receiveData = new byte[1024];

String sentence = inFromUser.readLine();

sendData = sentence.getBytes();

DatagramPacketsendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 9876);

clientSocket.send(sendPacket);

DatagramPacketreceivePacket = new DatagramPacket(receiveData, receiveData.length);

clientSocket.receive(receivePacket);

String modifiedSentence = new String(receivePacket.getData());

System.out.println("FROM SERVER:" + modifiedSentence);

clientSocket.close();

}

}

/\*\*\* B4.B. UDP chat p2p server \*\*\*/

import java.io.\*;

import java.net.\*;

class UDPServer

{

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

{

DatagramSocketserverSocket = new DatagramSocket(9876);

byte[] receiveData = new byte[1024];

byte[] sendData = new byte[1024];

while(true)

{

DatagramPacketreceivePacket = new DatagramPacket(receiveData, receiveData.length);

serverSocket.receive(receivePacket);

String sentence = new String( receivePacket.getData());

System.out.println("RECEIVED: " + sentence);

InetAddressIPAddress = receivePacket.getAddress();

int port = receivePacket.getPort();

String capitalizedSentence = sentence.toUpperCase();

sendData = capitalizedSentence.getBytes();

DatagramPacketsendPacket =

new DatagramPacket(sendData, sendData.length, IPAddress, port);

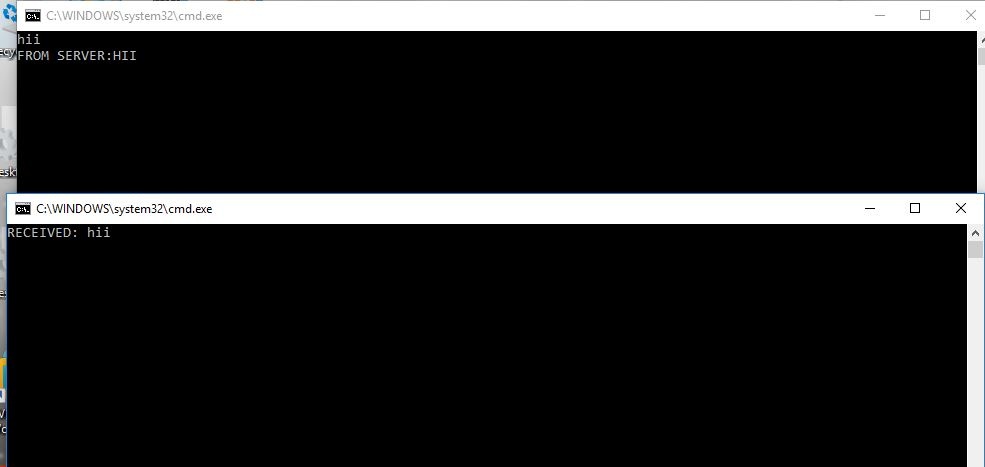
serverSocket.send(sendPacket);

}

}

}

\*\*\* OUTPUT of B4.B p2p\*\*\*



/\*\*\* B6.A.NS2. \*\*\*/

/\*\*\*\*\*A:----tcp\_udp\_ftp

#Create a simulator object

set ns [new Simulator]

#Define different colors for data flows (for NAM)

$ns color 1 Blue

$ns color 2 Red

#Open the NAM trace file

set nf [open out.nam w]

$ns namtrace-all $nf

#Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

#Close the NAM trace file

close $nf

#Execute NAM on the trace file

exec namout.nam&

exit 0

}

#Create four nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

#Create links between the nodes

$ns duplex-link $n0 $n2 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 2Mb 10ms DropTail

$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10

$ns queue-limit $n2 $n3 10

#Give node position (for NAM)

$ns duplex-link-op $n0 $n2 orient right-down

$ns duplex-link-op $n1 $n2 orient right-up

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

#Monitor the queue for link (n2-n3). (for NAM)

$ns duplex-link-op $n2 $n3 queuePos 0.5

#Setup a TCP connection

set tcp [new Agent/TCP]

$tcp set class\_ 1

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n3 $sink

$ns connect $tcp $sink

#Setup a FTP over TCP connection

set ftp [new Application/FTP]

$ftp attach-agent $tcp

#Setup a UDP connection

set udp [new Agent/UDP]

$udp set class\_ 2

$ns attach-agent $n1 $udp

set null [new Agent/Null]

$ns attach-agent $n3 $null

$ns connect $udp $null

#Setup a CBR over UDP connection

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set type\_ CBR

$cbr set packet\_size\_ 1000

#Schedule events for the CBR and FTP agents

$ns at 0.1 "$cbr start"

$ns at 1.0 "$ftp start"

$ns at 4.0 "$ftp stop"

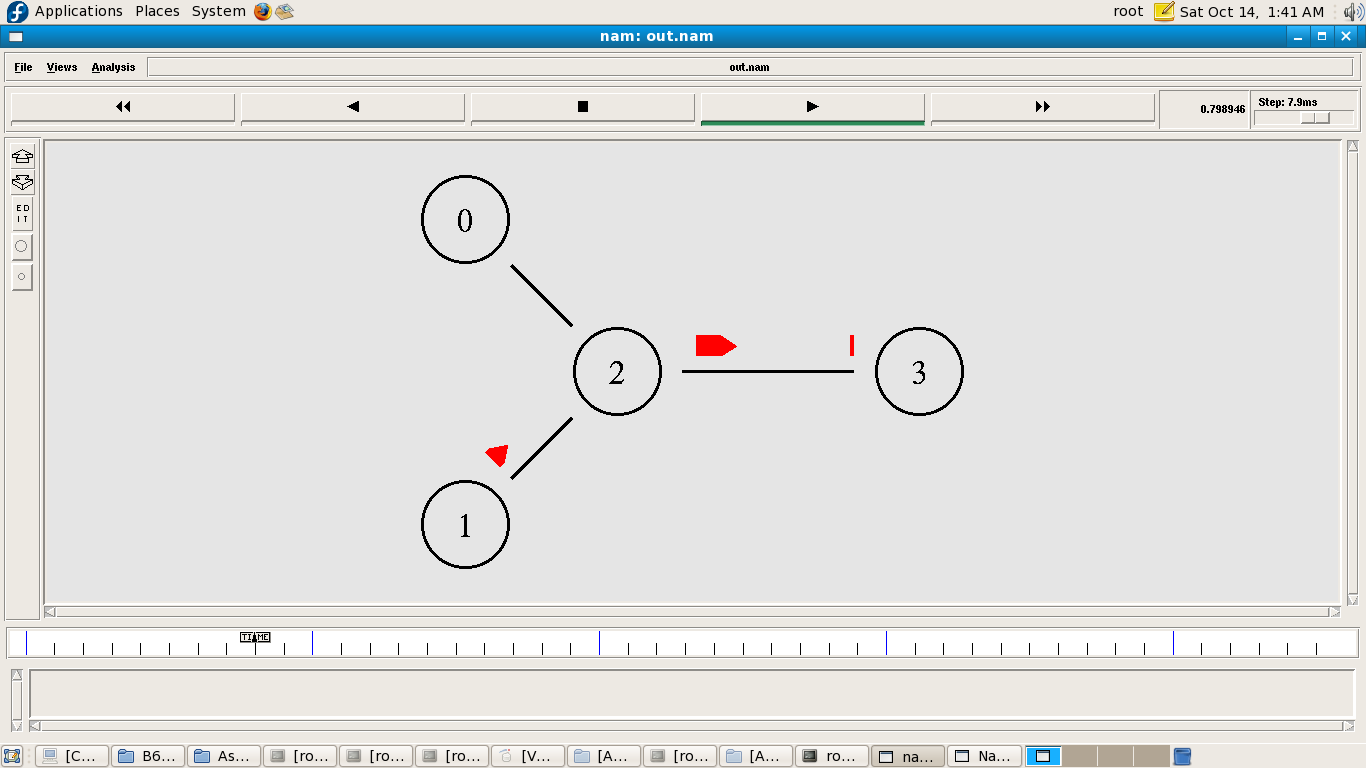
$ns at 4.5 "$cbr stop"

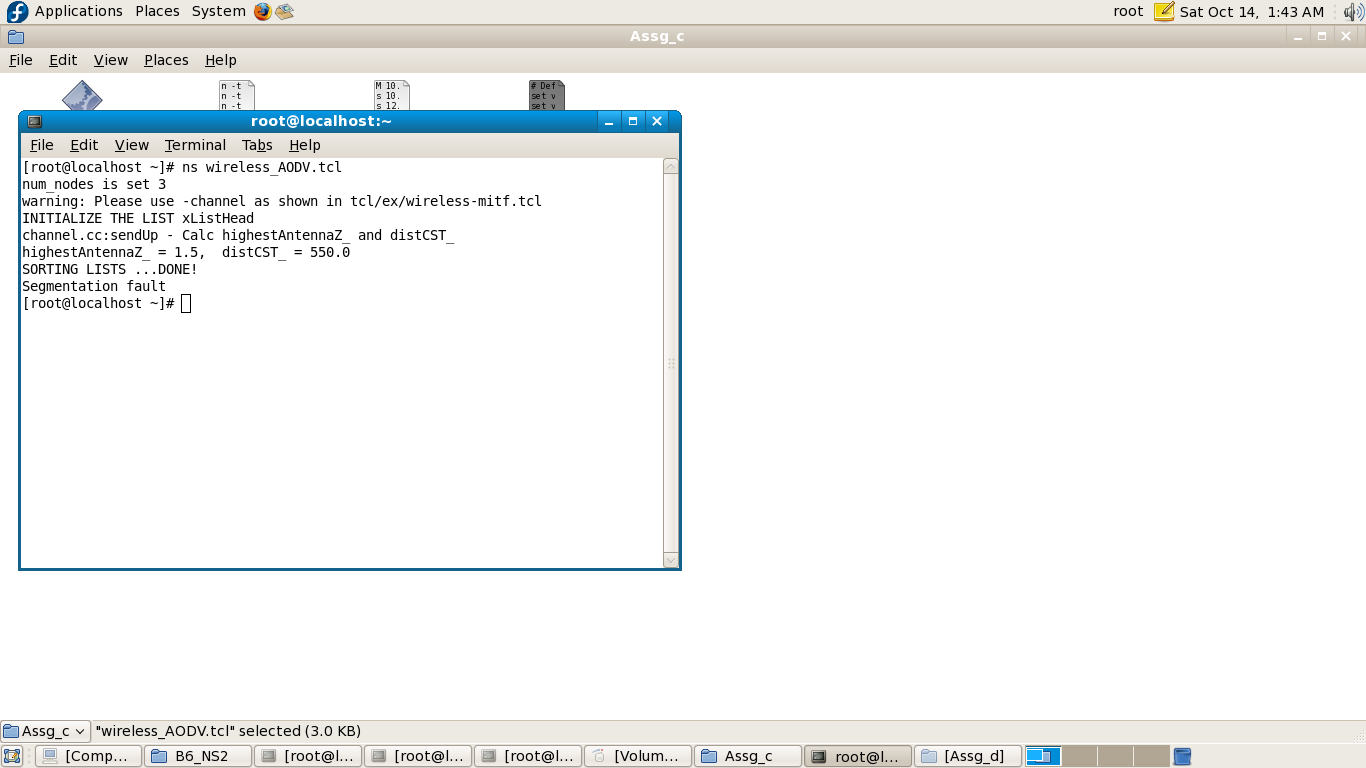
#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns run





/\*\*\* B6.A.NS2. \*\*\*/

/\*\*\*\*\*C:----wireless\_AODV

# Define options

set val(chan) Channel/WirelessChannel ;# channel type

set val(prop) Propagation/TwoRayGround ;# radio-propagation model

set val(netif) Phy/WirelessPhy ;# network interface type

set val(mac) Mac/802\_11 ;# MAC type

set val(ifq) Queue/DropTail/PriQueue ;# interface queue type

set val(ll) LL ;# link layer type

set val(ant) Antenna/OmniAntenna ;# antenna model

set val(ifqlen) 50 ;# max packet in ifq

set val(nn) 3 ;# number of mobilenodes

set val(rp) AODV ;# routing protocol

set val(x) 500 ;# X dimension of topography

set val(y) 400 ;# Y dimension of topography

set val(stop) 110 ;# time of simulation end

set ns [new Simulator]

set tracefd [open wireless.tr w]

set namtrace [open wireless.nam w]

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object

set topo [new Topography]

$topo load\_flatgrid $val(x) $val(y)

create-god $val(nn)

#

# Create nnmobilenodes [$val(nn)] and attach them to the channel.

#

# configure the nodes

$ns node-config -adhocRouting $val(rp) \

-llType $val(ll) \

-macType $val(mac) \

-ifqType $val(ifq) \

-ifqLen $val(ifqlen) \

-antType $val(ant) \

-propType $val(prop) \

-phyType $val(netif) \

-channelType $val(chan) \

-topoInstance $topo \

-agentTrace OFF \

-routerTrace OFF \

-macTrace ON \

-movementTrace ON

for {set i 0} {$i< $val(nn) } { incri } {

set node\_($i) [$ns node]

}

# Provide initial location of mobilenodes

$node\_(0) set X\_ 5.0

$node\_(0) set Y\_ 5.0

$node\_(0) set Z\_ 0.0

$node\_(1) set X\_ 490.0

$node\_(1) set Y\_ 285.0

$node\_(1) set Z\_ 0.0

$node\_(2) set X\_ 150.0

$node\_(2) set Y\_ 240.0

$node\_(2) set Z\_ 0.0

# Generation of movements

$ns at 10.0 "$node\_(0) setdest 250.0 250.0 3.0"

$ns at 15.0 "$node\_(1) setdest 45.0 285.0 5.0"

$ns at 110.0 "$node\_(0) setdest 480.0 300.0 5.0"

set tcp [new Agent/TCP]

$tcp set class\_ 2

set sink [new Agent/TCPSink]

$ns attach-agent $node\_(0) $tcp

$ns attach-agent $node\_(1) $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns at 10.0 "$ftp start"

# Define node initial position in nam

for {set i 0} {$i< $val(nn)} { incri } {

# 30 defines the node size for nam

$ns initial\_node\_pos $node\_($i) 50

}

# Telling nodes when the simulation ends

for {set i 0} {$i< $val(nn) } { incri } {

$ns at $val(stop) "$node\_($i) reset";

}

# ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

#$ns at 110.01 "puts \"end simulation\" ; $ns halt"

proc stop {} {

global ns tracefdnamtrace

$ns flush-trace

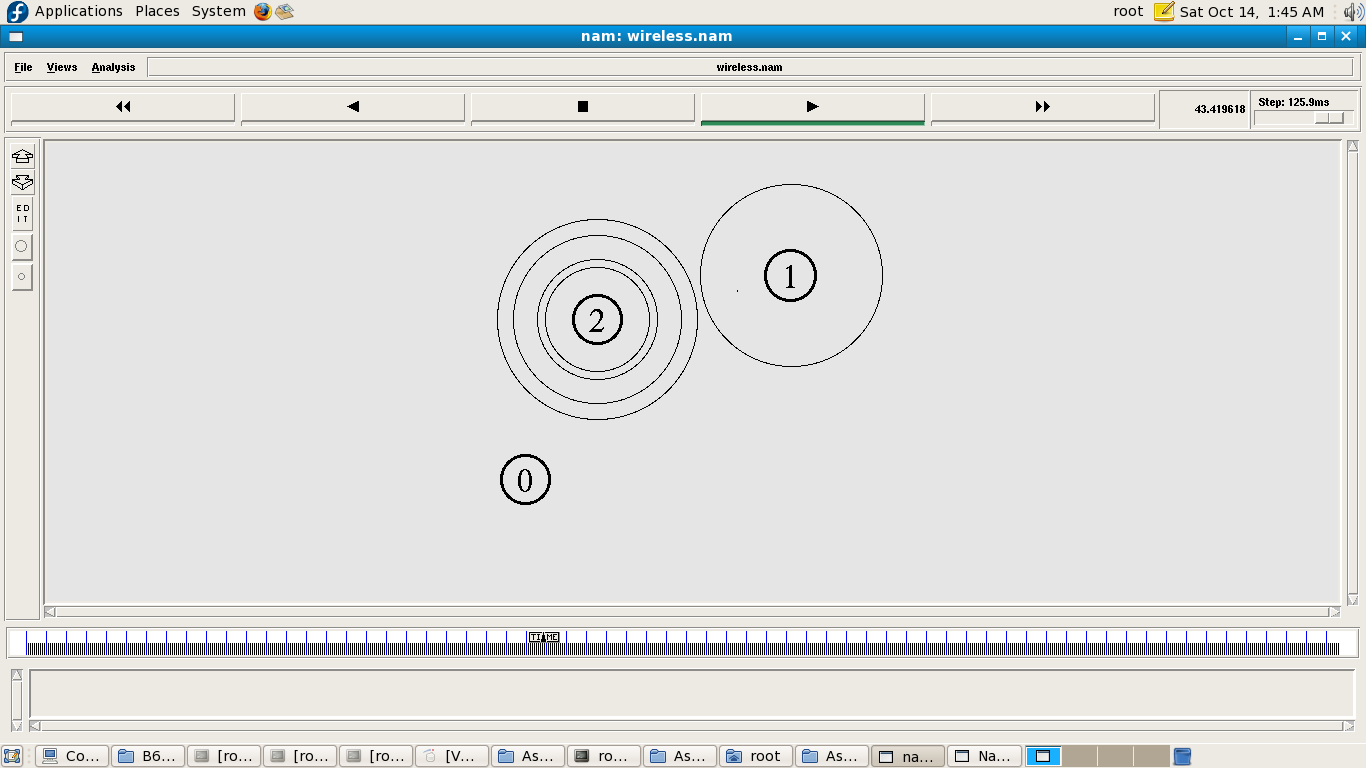
close $tracefd

close $namtrace

exec namwireless.nam&

}

$ns run



/\*\*\* B6.A.NS2. \*\*\*/

/\*\*\*\*\*D:----wired

set ns [new Simulator]

#Define different colors for data flows (for NAM)

$ns color 1 Blue

$ns color 2 Red

#Open the Trace files

set file1 [open out.tr w]

$ns trace-all $file1

#Open the NAM trace file

set file2 [open out.nam w]

$ns namtrace-all $file2

#Define a 'finish' procedure

proc finish {} {

global ns file1 file2

$ns flush-trace

close $file1

close $file2

exec namout.nam&

exit 0

}

#Create six nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$ns at 0.1 "$n1 label \"CBR\""

$ns at 1.0 "$n0 label \"FTP\""

#Create links between the nodes

$ns duplex-link $n0 $n2 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 2Mb 10ms DropTail

$ns duplex-link $n2 $n3 0.3Mb 100ms DropTail

$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail

$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail

#Give node position

$ns duplex-link-op $n0 $n2 orient right-down

$ns duplex-link-op $n1 $n2 orient right-up

$ns simplex-link-op $n2 $n3 orient right

$ns simplex-link-op $n3 $n2 orient left

$ns duplex-link-op $n3 $n4 orient right-up

$ns duplex-link-op $n3 $n5 orient right-down

#Set Queue Size of link (n2-n3) to 10

$ns queue-limit $n2 $n3 40

#Setup a TCP connection

set tcp [new Agent/TCP]

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n4 $sink

$ns connect $tcp $sink

$tcp set fid\_ 1

$tcp set window\_ 8000

$tcp set packetSize\_ 552

#Setup a FTP over TCP connection

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ftp set type\_ FTP

#Setup a UDP connection

set udp [new Agent/UDP]

$ns attach-agent $n1 $udp

set null [new Agent/Null]

$ns attach-agent $n5 $null

$ns connect $udp $null

$udp set fid\_ 2

#Setup a CBR over UDP connection

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set type\_ CBR

$cbr set packet\_size\_ 1000

$cbr set rate\_ 0.01mb

$cbr set random\_ false

$ns at 0.1 "$cbr start"

$ns at 1.0 "$ftp start"

$ns at 124.0 "$ftp stop"

$ns at 624.5 "$cbr stop"

# Trace Congestion Window and RTT

set file [open cwnd\_rtt.tr w]

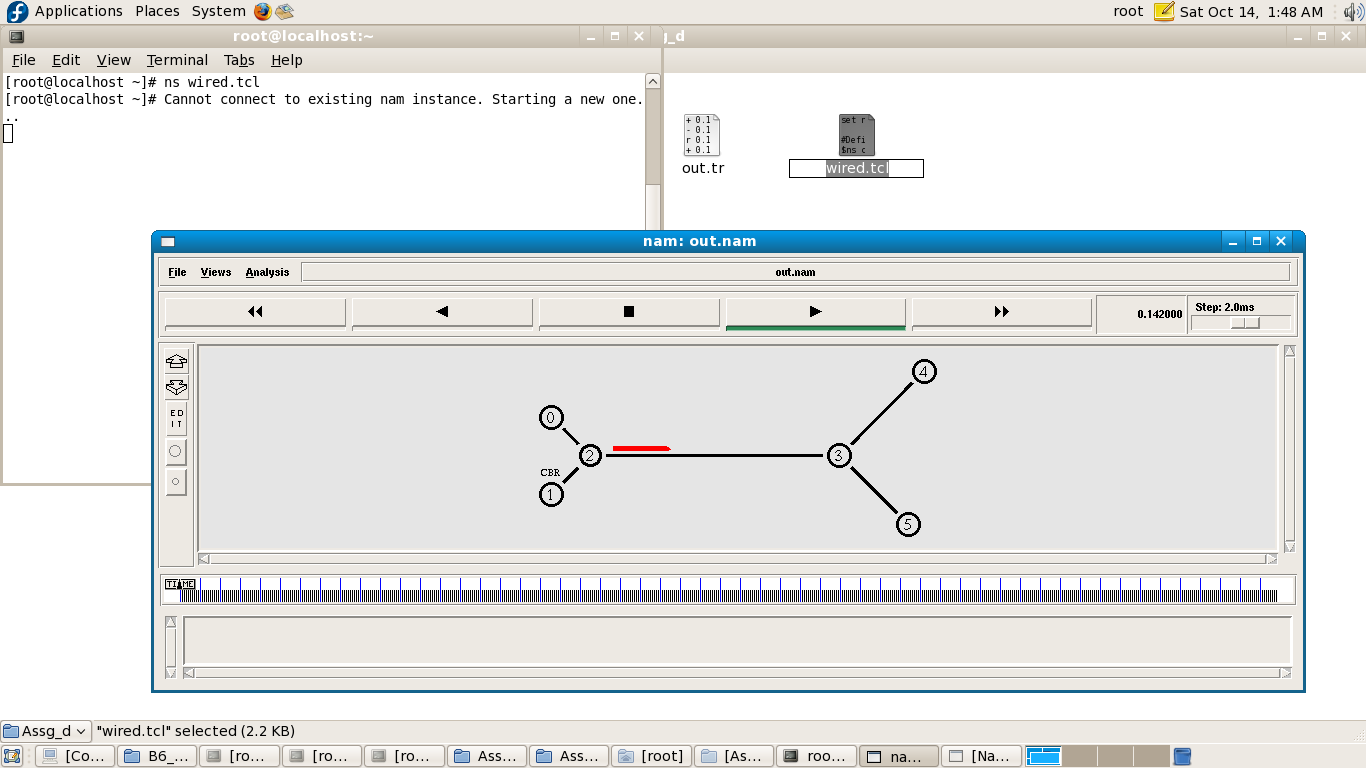
$tcp attach $file

$tcp trace cwnd\_

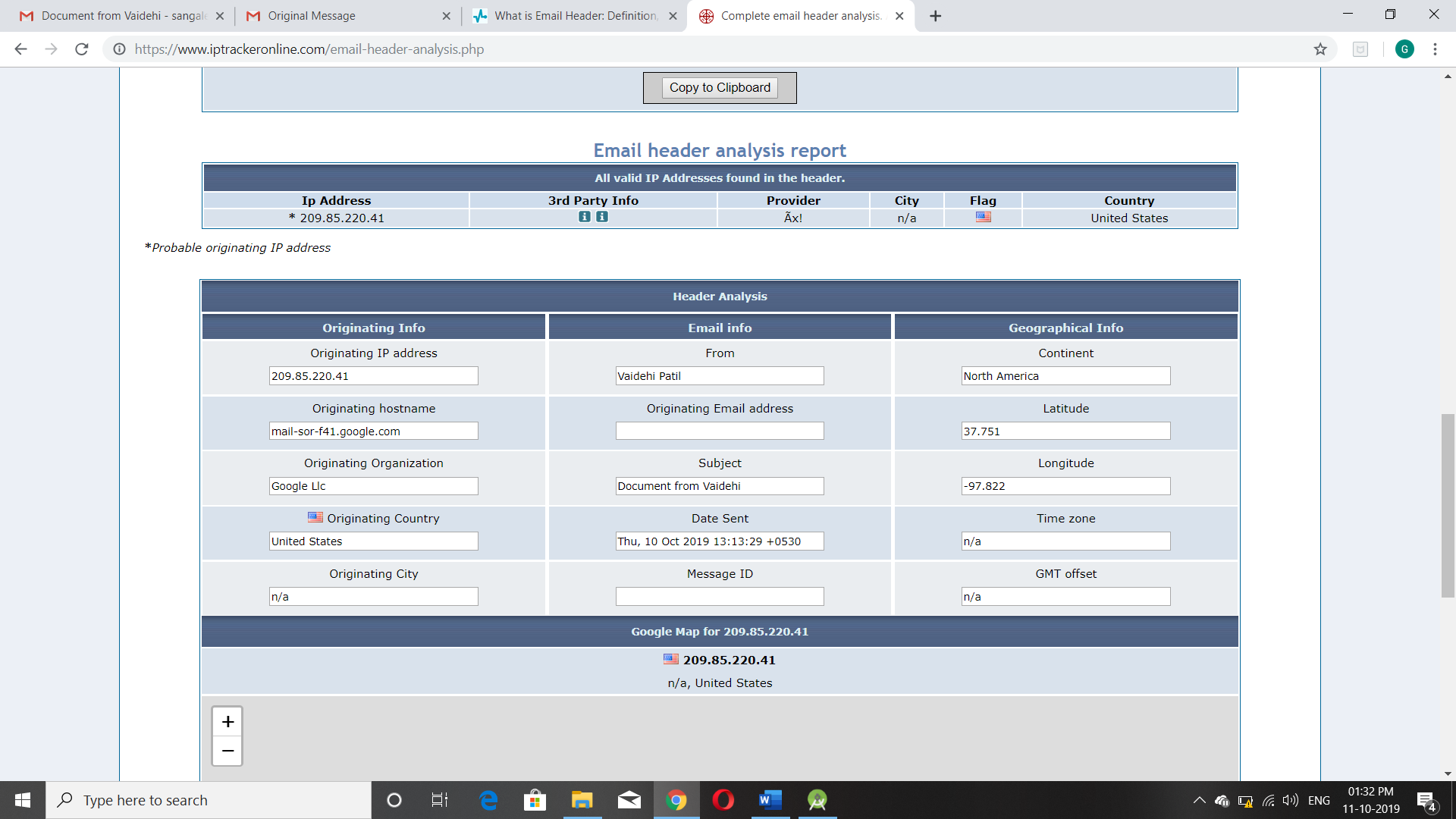
$tcp trace rtt\_

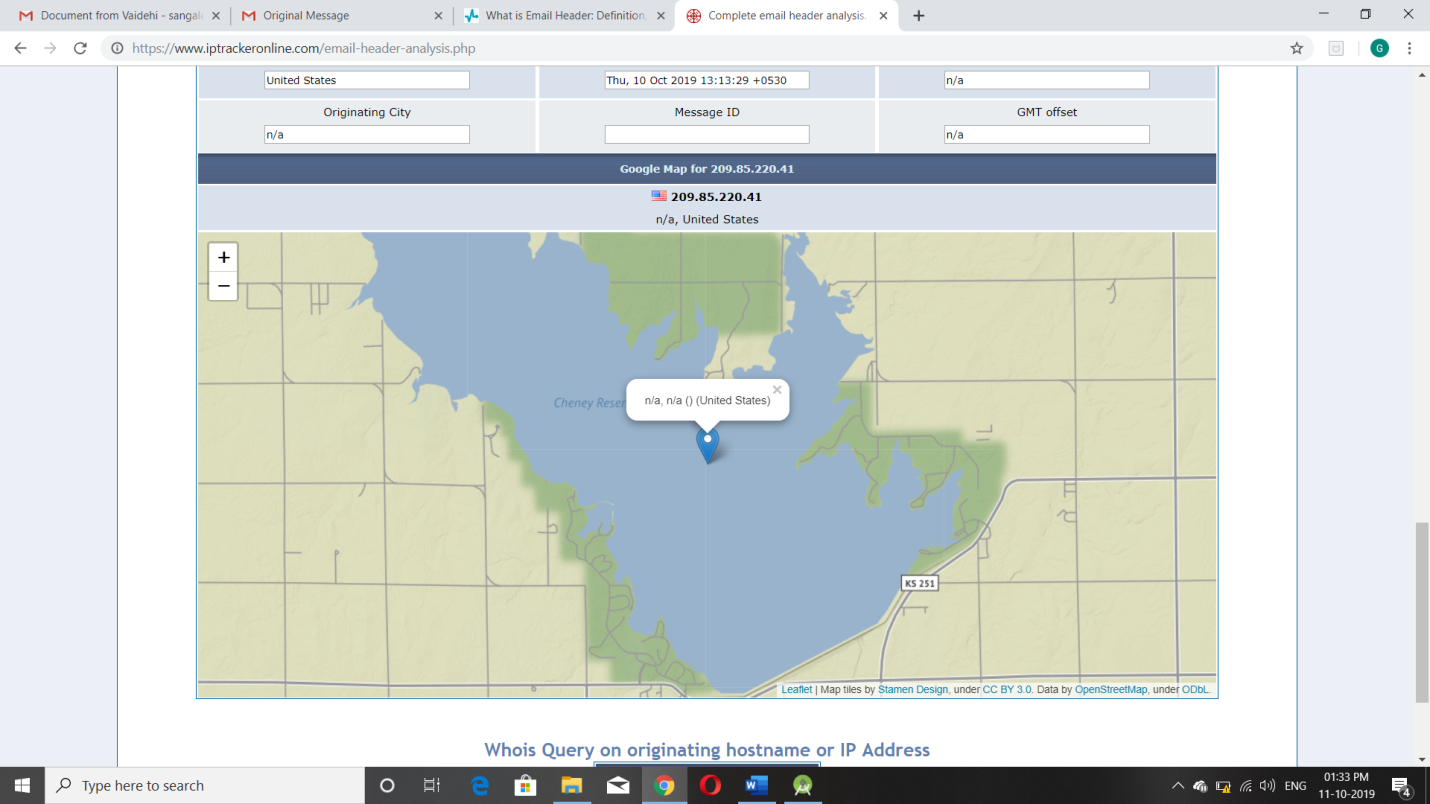
$ns at 625.0 "finish"

$ns run

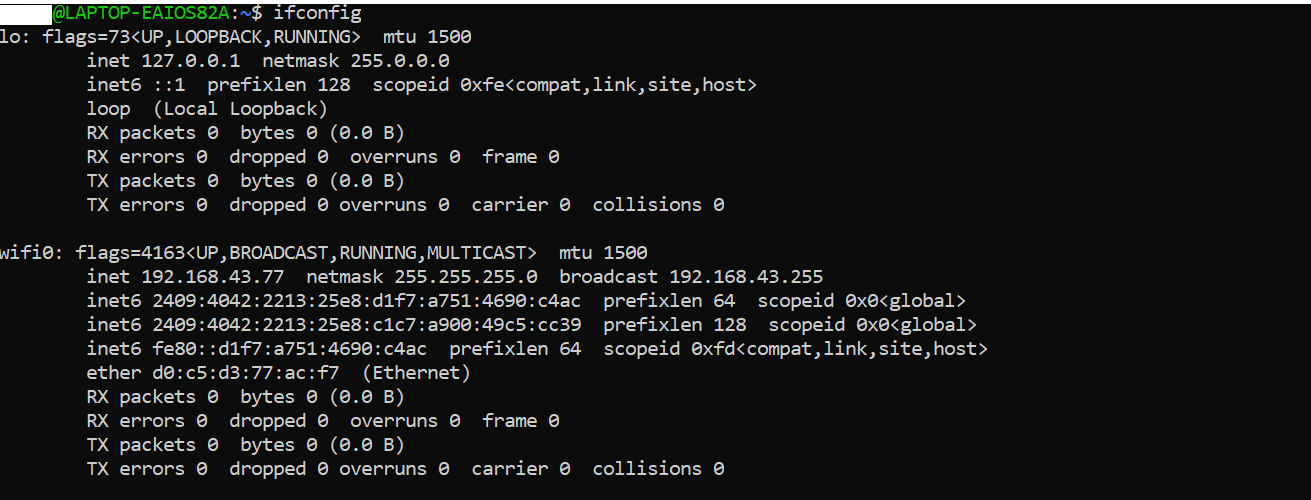


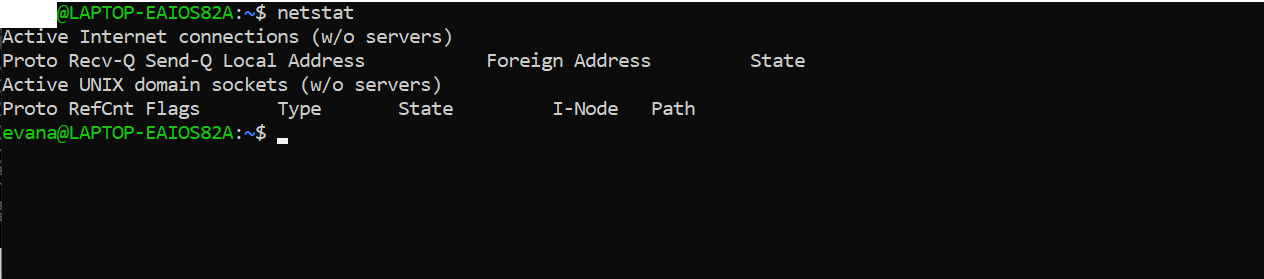
Bridge the gap

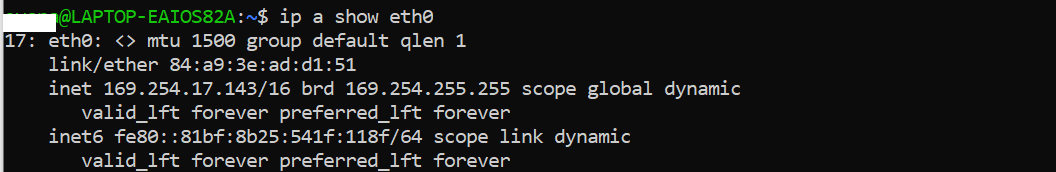




LINUX COMMANDS







IP TABLE (LINUX FIREWALL)

# /etc/init.d/iptables start

# /etc/init.d/iptables stop

# /etc/init.d/iptables restart

#chkconfig --level 345 iptables on

#service iptables save

[root@tecmint ~]# iptables -L -n -v

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)

pkts bytes target prot opt in out source destination

6 396 ACCEPT all -- \* \* 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED

0 0 ACCEPT icmp -- \* \* 0.0.0.0/0 0.0.0.0/0

0 0 ACCEPT all -- lo \* 0.0.0.0/0 0.0.0.0/0

0 0 ACCEPT tcp -- \* \* 0.0.0.0/0 0.0.0.0/0 state NEW tcp dpt:22

0 0 REJECT all -- \* \* 0.0.0.0/0 0.0.0.0/0 reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)

pkts bytes target prot opt in out source destination

0 0 REJECT all -- \* \* 0.0.0.0/0 0.0.0.0/0 reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 5 packets, 588 bytes)

pkts bytes target prot opt in out source destination

[root@tecmint ~]# iptables -n -L -v --line-numbers

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)

num pkts bytes target prot opt in out source destination

1 51 4080 ACCEPT all -- \* \* 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED

2 0 0 ACCEPT icmp -- \* \* 0.0.0.0/0 0.0.0.0/0

3 0 0 ACCEPT all -- lo \* 0.0.0.0/0 0.0.0.0/0

4 0 0 ACCEPT tcp -- \* \* 0.0.0.0/0 0.0.0.0/0 state NEW tcp dpt:22

5 0 0 REJECT all -- \* \* 0.0.0.0/0 0.0.0.0/0 reject-with icmp-host-prohibited

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)

num pkts bytes target prot opt in out source destination

1 0 0 REJECT all -- \* \* 0.0.0.0/0 0.0.0.0/0 reject-with icmp-host-prohibited

Chain OUTPUT (policy ACCEPT 45 packets, 5384 bytes)

num pkts bytes target prot opt in out source destination

[root@tecmint ~]# iptables -F

[root@tecmint ~]# iptables -L **INPUT** -n --line-numbers

Chain INPUT (policy ACCEPT)

num target prot opt source destination

1 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 state RELATED,ESTABLISHED

2 ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0

3 ACCEPT all -- 0.0.0.0/0 0.0.0.0/0

4 ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 state NEW tcp dpt:22

5 REJECT all -- 0.0.0.0/0 0.0.0.0/0 reject-with icmp-host-prohibited

[root@tecmint ~]# iptables -L **OUTPUT** -n --line-numbers

Chain OUTPUT (policy ACCEPT)

num target prot opt source destination

[root@tecmint ~]# iptables -D INPUT 5

[root@tecmint ~]# iptables -I INPUT 5 -s ipaddress -j DROP