DATE:12/04/2022

USN:1NT19IS162

NP LAB

PART A

1.Design a c program in which the sender module should count the number of bytes in the frame and receiver module should display each frame it received.

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

char res[100];

void sender(){

int n,i,len;

char frm[100],l[100];

printf("Enter the number of frames\n");

scanf("%d",&n);

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

printf("Enter the frame %d\n",i+1);

scanf("%s",&frm);

len=strlen(frm);

sprintf(l,"%d",len);

strcat(l,frm);

strcat(res,l);

}

printf("The message is %s\n",res);

}

void receiver(){

int len,i,j;

printf("Received frame \n");

for(i=0;i<strlen(res);i++){

len=res[i]-'0';

for(j=i+1;j<=(i+len);j++)

printf("%c",res[j]);

i=i+len;

printf("\n");

}

}

void main(){

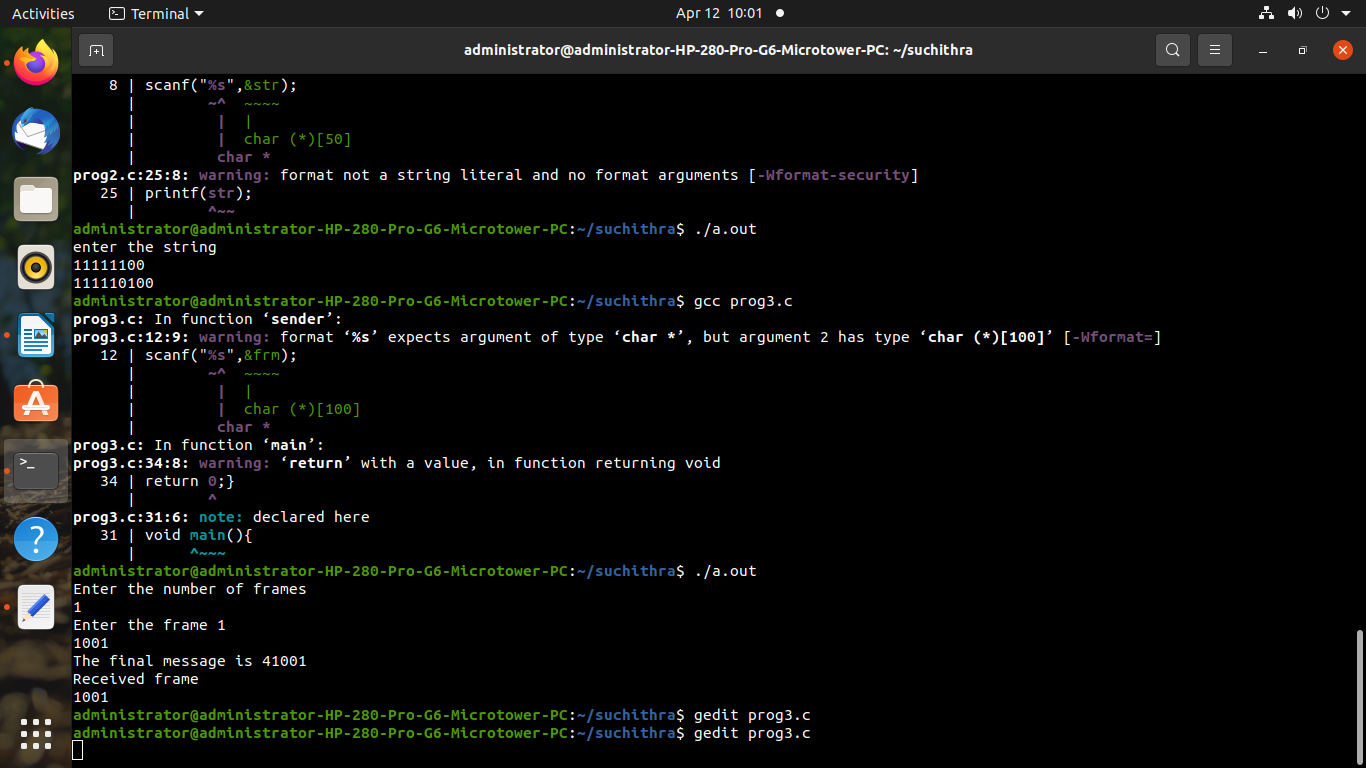
sender();

receiver();

return 0;

}

OUTPUT:



2.Design a c program to implement bit stuffing concept in data link layer.

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

void receiver(int \*frame,int l2){

int i,j,counter,l3;

int msg[100];

int data[100];

l3=l2-8;

j=0;

for(i=8;i<l3;i++){

if(counter==5){

i++;

counter=0;

}

if(frame[i]==0){

msg[j]=frame[i];

j++;

counter=0;

}

else{

msg[j]=frame[i];

j++;

counter++;

}

}

printf("\nReceived message is: \n");

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

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

printf("\n");

}

void sender(){

int data[100],frame[100],framelen=0,n,i,j=8;

int count,zeroadded=0,zero;

printf("Enter the number of bits\n");

scanf("%d",&n);

printf("Enter data for bits\n");

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

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

frame[0]=0;

frame[1]=1;

frame[2]=1;

frame[3]=1;

frame[4]=1;

frame[5]=1;

frame[6]=1;

frame[7]=0;

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

if((count==5)&&(zero==1)){

frame[j++]=0;

zeroadded++;

count=0;

}

if(data[i]==0){

frame[j++]=data[i];

count=0;

zero=1;

}

else{

frame[j++]=data[i];

count++;

}}

if(count==5){

frame[j++]=0;

zeroadded++;

count=0;

}

frame[j++]=0;

frame[j++]=1;

frame[j++]=1;

frame[j++]=1;

frame[j++]=1;

frame[j++]=1;

frame[j++]=1;

frame[j++]=0;

framelen=n+16+zeroadded;

printf("length of frame sent %d \n",framelen);

printf("Frame sent: ");

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

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

}

receiver(frame,framelen);

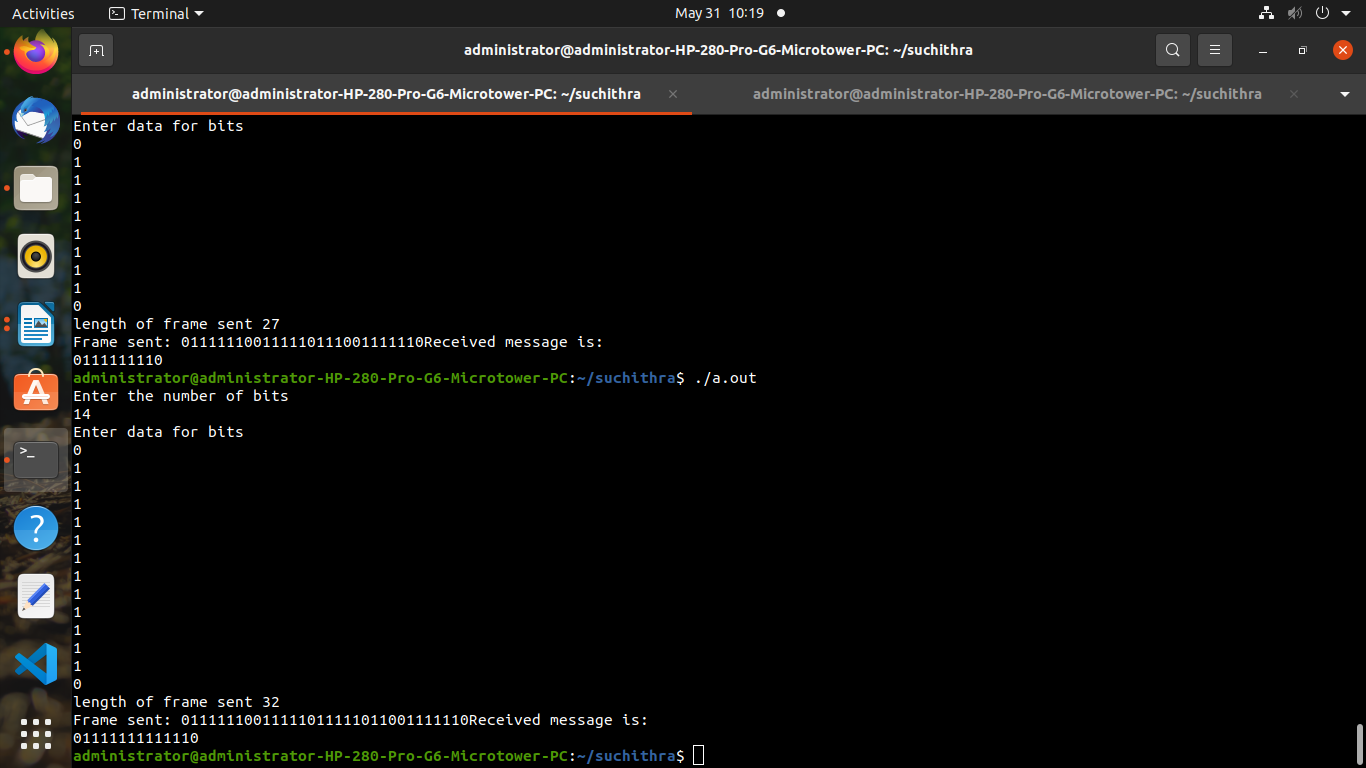
}

void main(){

sender();

}

OUTPUT:



3.Design and implement CRC error detection method using data link layer.

#include<stdio.h>

#include<string.h>

#include<unistd.h>

#include<stdlib.h>

#define gen strlen(g)

int a,c,e;

char t[128],cs[128],g[]="1101";

void xor(){

for(c=1;c<gen;c++)

cs[c]=((cs[c]==g[c])?'0':'1');

}

void crc(){

for(e=0;e<gen;e++)

cs[e]=t[e];

do{

if(cs[0]=='1')

xor();

for(c=0;c<gen-1;c++)

cs[c]=cs[c+1];

cs[c]=t[e++];

}

while(e<=a+gen-1);

}

int main(){

printf("Enter ploynomial data:");

scanf("%s",&t);

printf("\nGenerator polynomial : %s\n",g);

a=strlen(t);

for(e=a;e<a+gen-1;e++)

t[e]='0';

printf("Modified data: %s \n",t);

crc();

printf("Checksum: %s \n",cs);

for(e=a;e<a+gen-1;e++)

t[e]=cs[e-a];

printf("Final codeword: %s \n",t);

printf("Test error detection 0(Yes) 1(No)?\n");

scanf("%d",&e);

if(e==0){

do{

printf("Enter the position where error has to be inserted \n");

scanf("%d",&e);

}

while(e==0 || e>a+gen-1);

t[e-1]=(t[e-1]=='0')?'1':'0';

printf("erroneous data %s\n",t);

}

crc();

for(e=0;(e<gen-1) && (cs[e]!='1');e++);

if(e<gen-1)

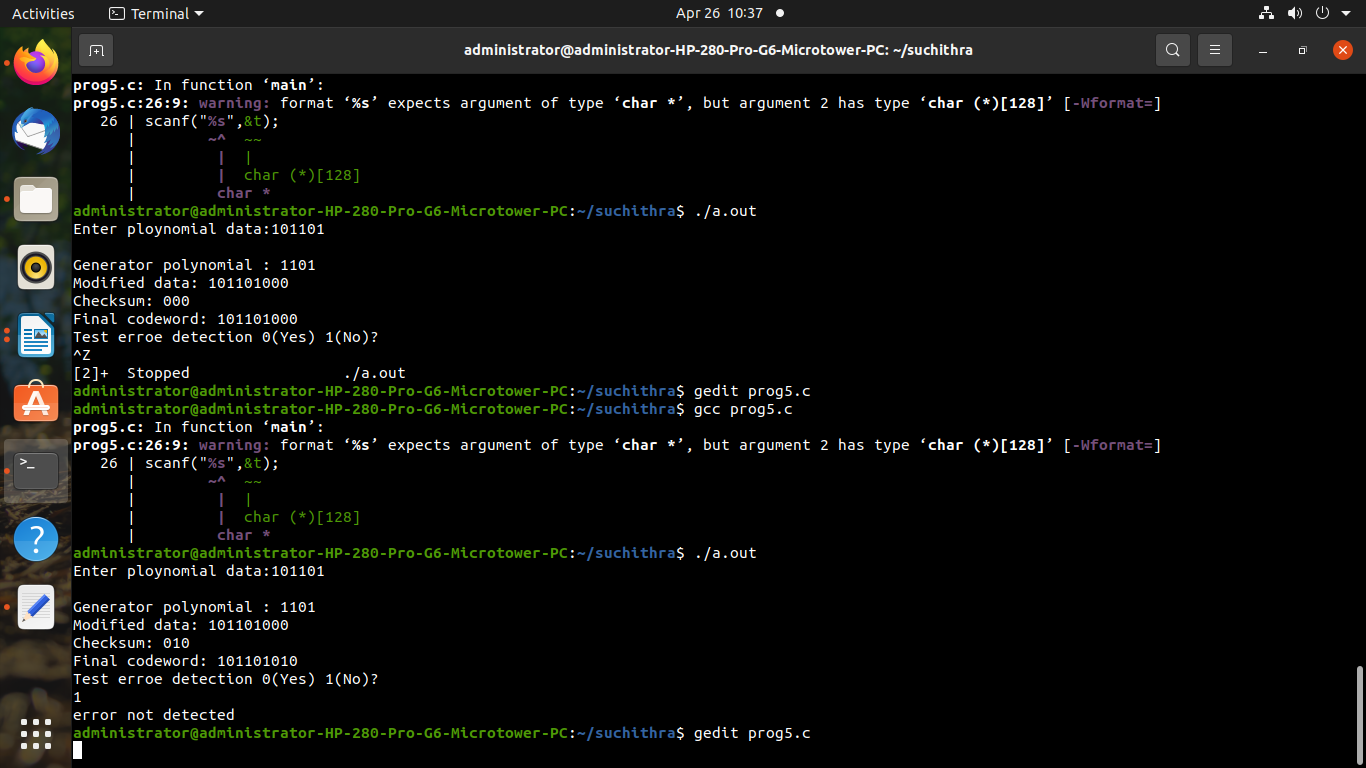
printf("error detected\n");

else

printf("error not detected\n");

}

OUTPUT:



TCP/UDP

UDP SERVER

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

int main(){

int udpSocket, nBytes;

char buffer[1024];

struct sockaddr\_in serverAddr, clientAddr;

struct sockaddr\_storage serverStorage;

socklen\_t addr\_size, client\_addr\_size;

int i;

/\*Create UDP socket\*/

udpSocket = socket(PF\_INET, SOCK\_DGRAM, 0);

/\*Configure settings in address struct\*/

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(8893);

serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

memset(serverAddr.sin\_zero, '\0', sizeof serverAddr.sin\_zero);

/\*Bind socket with address struct\*/

bind(udpSocket, (struct sockaddr \*) &serverAddr, sizeof(serverAddr));

/\*Initialise size variable to be used later on\*/

addr\_size = sizeof serverStorage;

while(1){

/\* Try to receive any incoming UDP datagram. Address and port of

\* requesting client will be stored on serverStorage variable \*/

nBytes = recvfrom(udpSocket,buffer,1024,0,(struct sockaddr \*)&serverStorage, &addr\_size);

/\*Convert message received to uppercase\*/

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

buffer[i] = toupper(buffer[i]);

/\*Send uppercase message back to client, using serverStorage as the address\*/

sendto(udpSocket,buffer,nBytes,0,(struct sockaddr \*)&serverStorage,addr\_size);

}

return 0;

}

UDP CLIENT

#include <stdio.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

int main(){

int clientSocket, portNum, nBytes;

char buffer[1024];

struct sockaddr\_in serverAddr;

socklen\_t addr\_size;

/\*Create UDP socket\*/

clientSocket = socket(PF\_INET, SOCK\_DGRAM, 0);

/\*Configure settings in address struct\*/

serverAddr.sin\_family = AF\_INET;

serverAddr.sin\_port = htons(8893);

serverAddr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

memset(serverAddr.sin\_zero, '\0', sizeof serverAddr.sin\_zero);

/\*Initialize size variable to be used later on\*/

addr\_size = sizeof serverAddr;

while(1){

printf("Type a sentence to send to server:\n");

fgets(buffer,1024,stdin);

printf("You typed: %s",buffer);

nBytes = strlen(buffer) + 1;

/\*Send message to server\*/

sendto(clientSocket,buffer,nBytes,0,(struct sockaddr \*)&serverAddr,addr\_size);

/\*Receive message from server\*/

nBytes = recvfrom(clientSocket,buffer,1024,0,NULL, NULL);

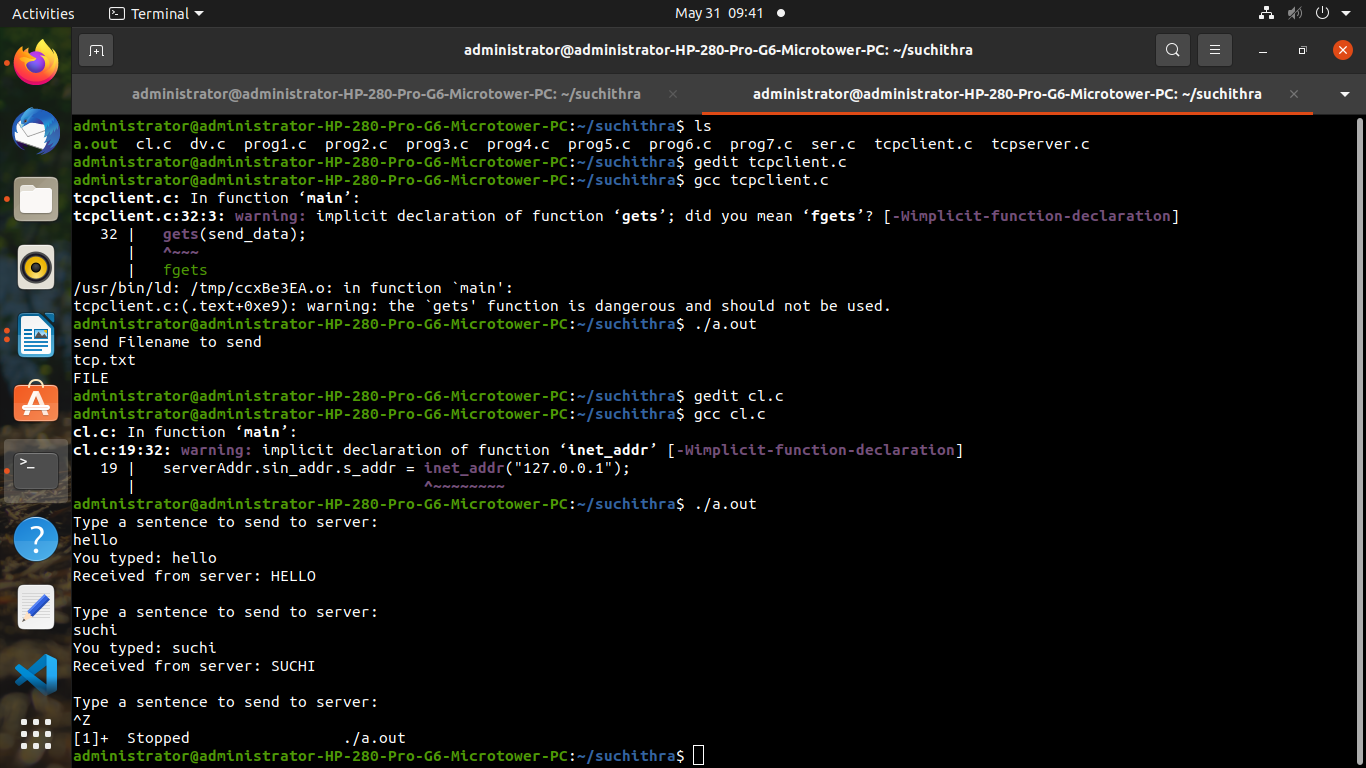
printf("Received from server: %s\n",buffer);

}

return 0;

}

OUTPUT:



TCP SERVER

#include<stdio.h>

#include<stdlib.h>

#include<arpa/inet.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<errno.h>

#include<unistd.h>

#include<netinet/in.h>

#include<string.h>

int main()

{

struct sockaddr\_in server\_addr;

struct sockaddr\_in client\_addr;

FILE \*fptr;

int sock,connected,bytes\_recv;

char ch,send\_data[1024],recv\_data[1024];

int sin\_size,flag = 0;

if((sock=socket(AF\_INET,SOCK\_STREAM,0))==-1)

{

perror("socket");

exit(1);

}

server\_addr.sin\_family=AF\_INET;

server\_addr.sin\_port=htons(6119);

server\_addr.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

if(bind(sock,(struct sockaddr \*)&server\_addr, sizeof(struct sockaddr))==-1)

{

perror("unable to bind");

exit(1);

}

if(listen(sock,5)==-1)

{

perror("lsten");

exit(1);

}

printf("tcp server is waiting for client on port XXXX\n");

sin\_size=sizeof(struct sockaddr\_in);

connected=accept(sock,(struct sockaddr \*)&client\_addr,&sin\_size);

while(1)

{

bytes\_recv=recv(connected,recv\_data,1024,0);

recv\_data[bytes\_recv]='\0';

printf("reciecved data is %s\n\n\n",recv\_data);

fptr=fopen(recv\_data,"r");

if(fptr==NULL)

{

strcpy(send\_data,"FILE");

send(connected,send\_data,strlen(send\_data),0);

}

ch = fgetc(fptr);

while(ch != EOF)//this loop searches the for the current word

{

// fscanf(fptr,"%s",send\_data);

send\_data[flag] = ch;

flag++;

ch = fgetc(fptr);

//send(connected,send\_data,strlen(send\_data),0);

}

send(connected,send\_data,strlen(send\_data),0);

//send\_data[0] = 'q';

//strcpy(send\_data,"q");

//send(connected,send\_data,strlen(send\_data),0);

close(connected);

break;

}

}

TCP CLIENT

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<arpa/inet.h>

#include<netdb.h>

#include<netinet/in.h>

#include<errno.h>

#include<string.h>

int main()

{

int sock,bytes\_recv;

struct sockaddr\_in server\_addr;

char recv\_data[1024],send\_data[1024];

struct hostent \*host;

host=gethostbyname("127.0.0.1");

if((sock=socket(AF\_INET,SOCK\_STREAM,0))==-1)

{

perror("socket");

exit(1);

}

server\_addr.sin\_family=AF\_INET;

server\_addr.sin\_port=htons(6119);

server\_addr.sin\_addr.s\_addr=inet\_addr("127.0.0.1");

if(connect(sock,(struct sockaddr \*)&server\_addr,sizeof(struct sockaddr))==-1)

{

perror("connect");

exit(1);

}

printf("send Filename to send\n");

gets(send\_data);

if(strcmp(send\_data,"q")!=0)

send(sock,send\_data,strlen(send\_data),0);

while((bytes\_recv=recv(sock,recv\_data,1024,0))>0)

{

recv\_data[bytes\_recv]='\0';

//printf("%s\n\n", recv\_data);

//if(strcmp(recv\_data,"q")==0)

// {

// close(sock);

// break;

// }

printf("%s\n", recv\_data);

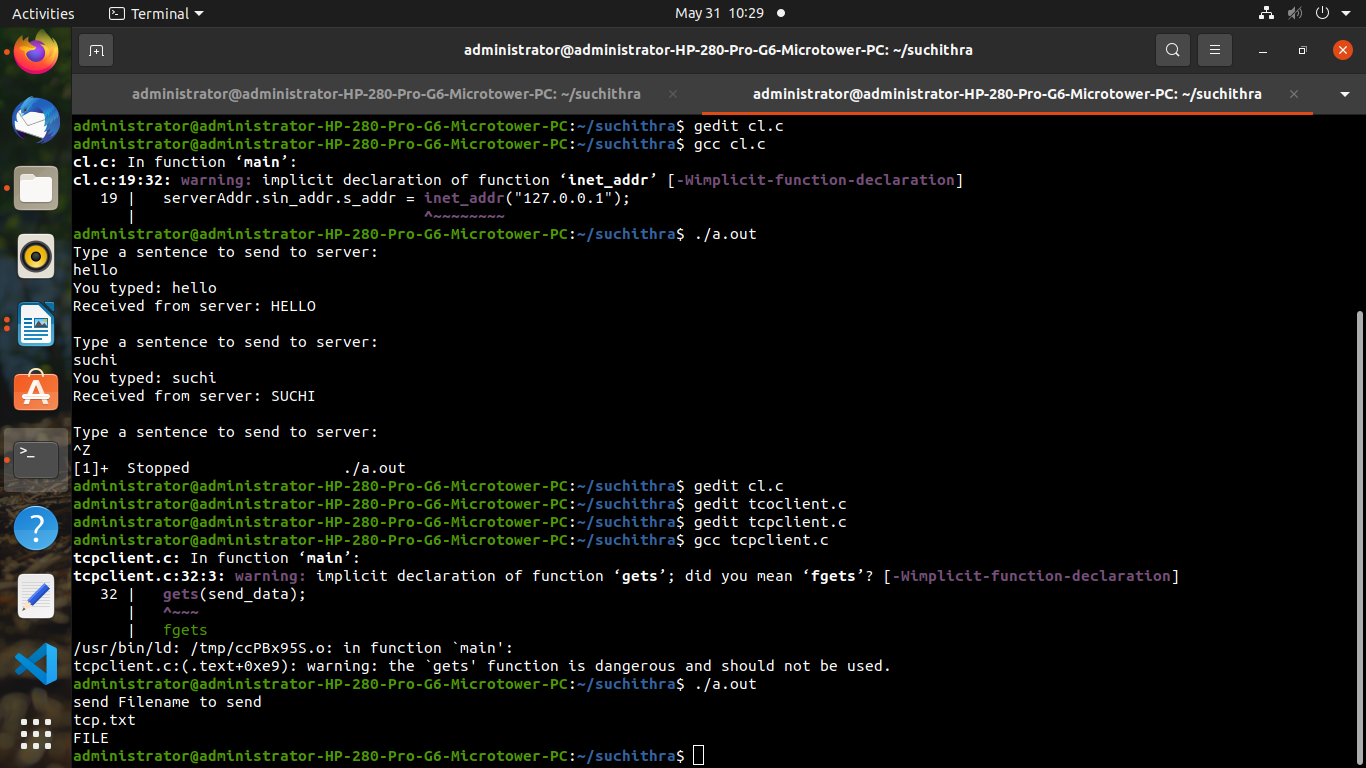
}

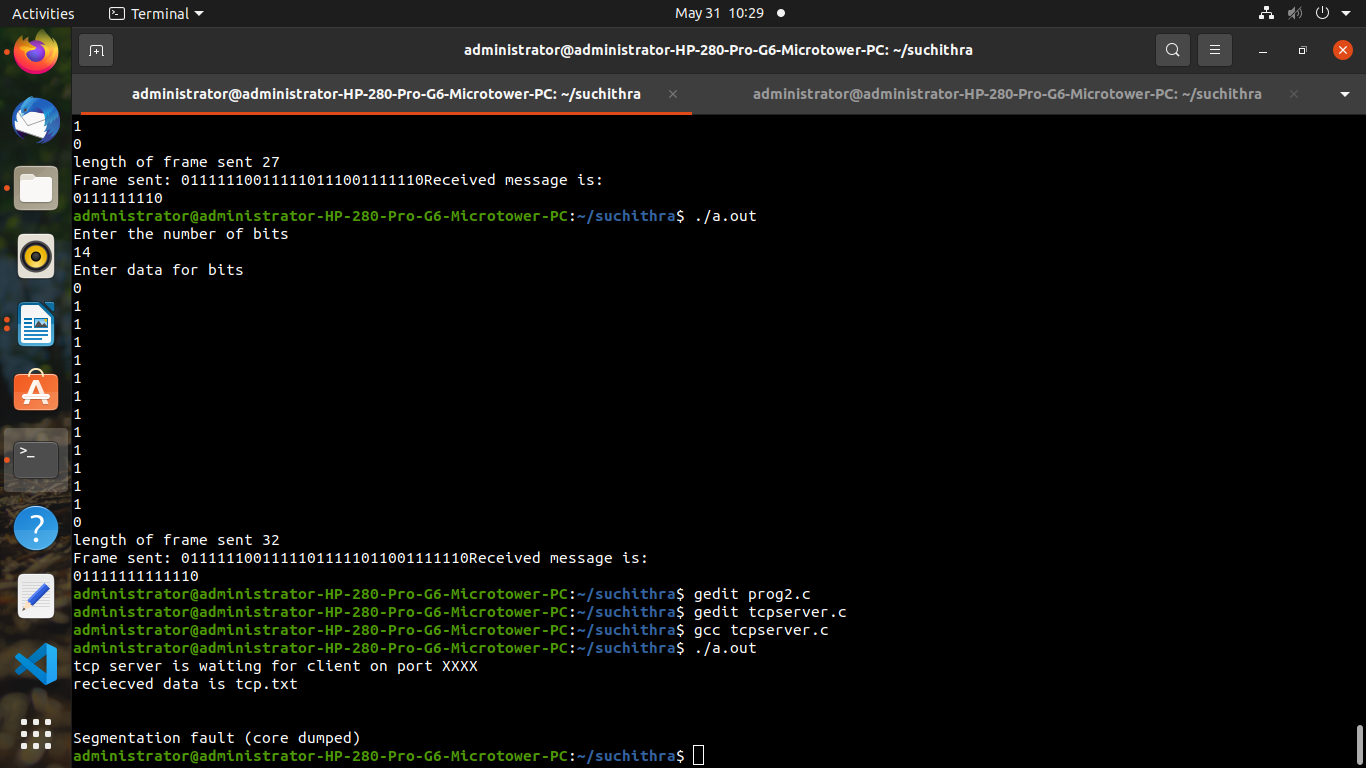
close(sock);

return 0;

}

OUTPUT:





DISTANCE VECTOR ROUTING

#include<stdio.h>

struct node

{

unsigned dist[20];

unsigned from[20];

}rt[10];

int main()

{

int dmat[20][20];

int n,i,j,k,count=0;

printf("enter the number of nodes\n");

scanf("%d",&n);

printf("enter the cost matrix\n");

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

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

{

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

dmat[i][i]=0;

rt[i].dist[j]=dmat[i][j];

rt[i].from[j]=j;

}

do{

count=0;

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

for(j=0;j<n;j++){

for(k=0;k<n;k++){

if(rt[i].dist[j]>dmat[i][k]+rt[k].dist[j]){

rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];

rt[i].from[j]=k;

count++;

}

}

}

}

}while(count!=0);

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

printf("\n\n state value for router %d is \n",i+1);

printf("\n Node \t via \t distance cost");

for(j=0;j<n;j++){

printf("\n %d \t %d \t %d",j+1,rt[i].from[j]+1,rt[i].dist[j]);

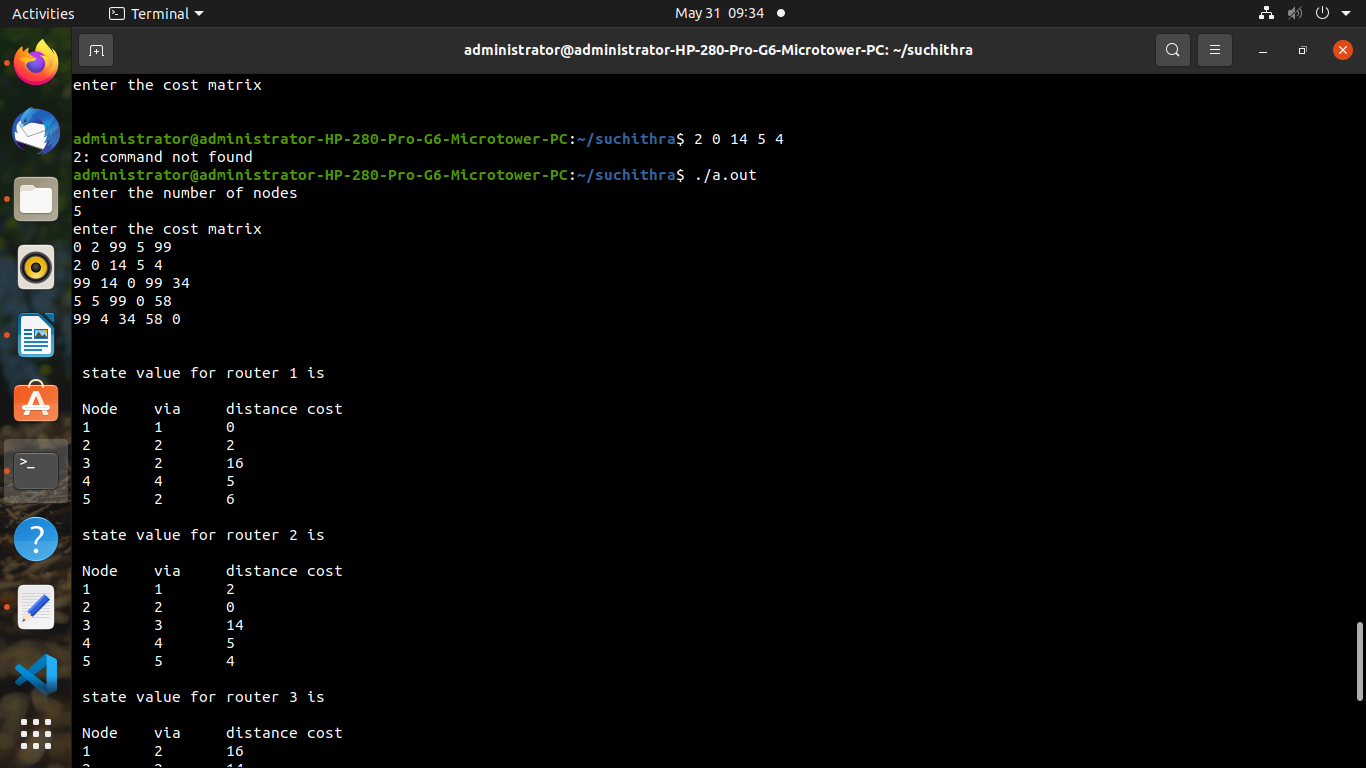
}

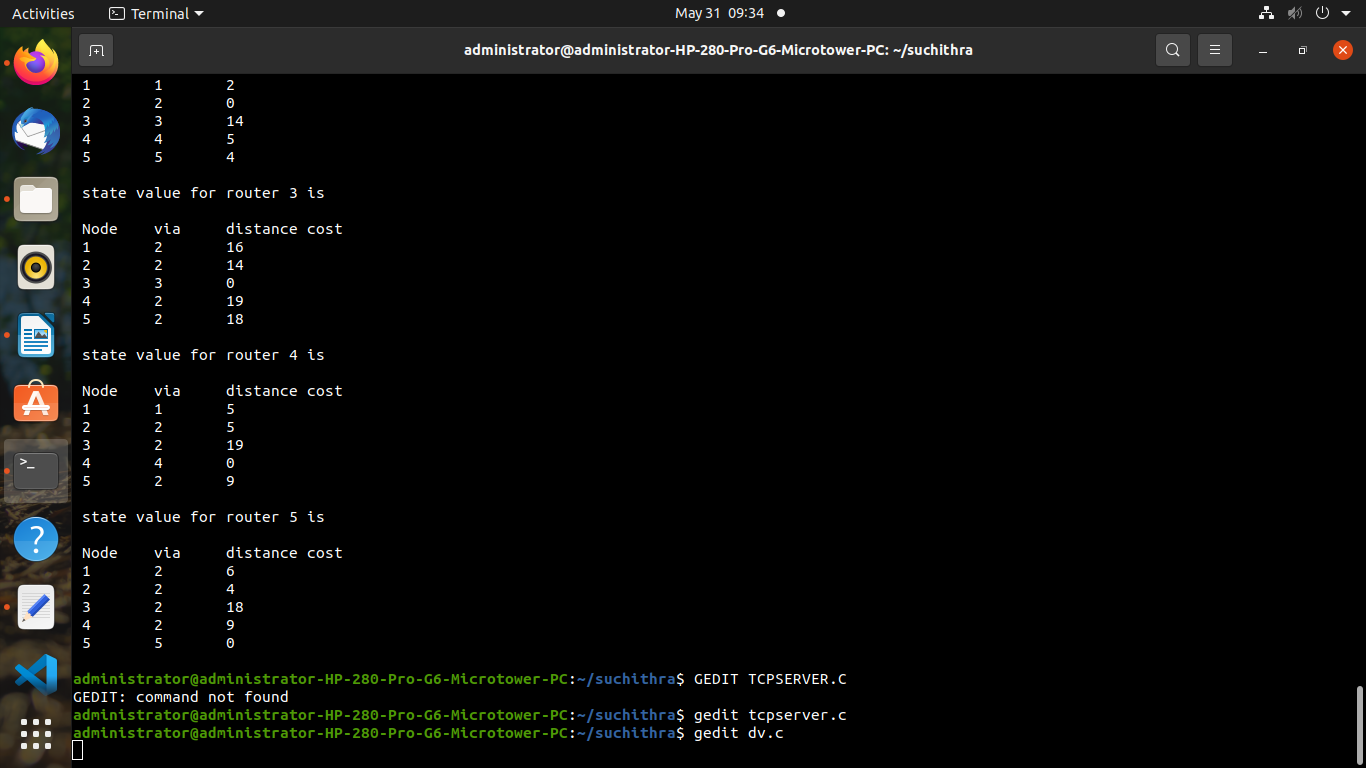
}

printf("\n\n");

}

OUTPUT:





CONGESTION CONTROL

#include<stdio.h>

#include<stdlib.h>

#define MIN(x,y) (x>y)?y:x

int main(){

int orate,drop=0,cap,x,count=0,inp[10]={0},nsec,ch,i=0;

printf("enter bucket size\n");

scanf("%d",&cap);

printf("enter the output rate\n");

scanf("%d",&orate);

do{

printf("\n enter number of packets coming at second %d :",i+1);

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

if(inp[i]>cap)

{

printf("Bucket overflow\n");

printf("Packet Discarded\n");

exit(0);

}

i++;

printf("\n enter 1 to contiue or 0 to quit..........");

scanf("%d",&ch);

}

while(ch);

nsec=i;

printf("\n Second \t Recieved \t Sent \t Dropped \tRemained \n");

for(i=0;count || i<nsec;i++)

{

printf(" %d",i+1);

printf(" \t\t%d\t ",inp[i]);

printf(" \t%d\t ",MIN((inp[i]+count),orate));

if((x=inp[i]+count-orate)>0)

{

if(x>cap)

{

count=cap;

drop=x-cap;

}

else

{

count=x;

drop=0;

}

}

else

{

drop=0;

count=0;

}

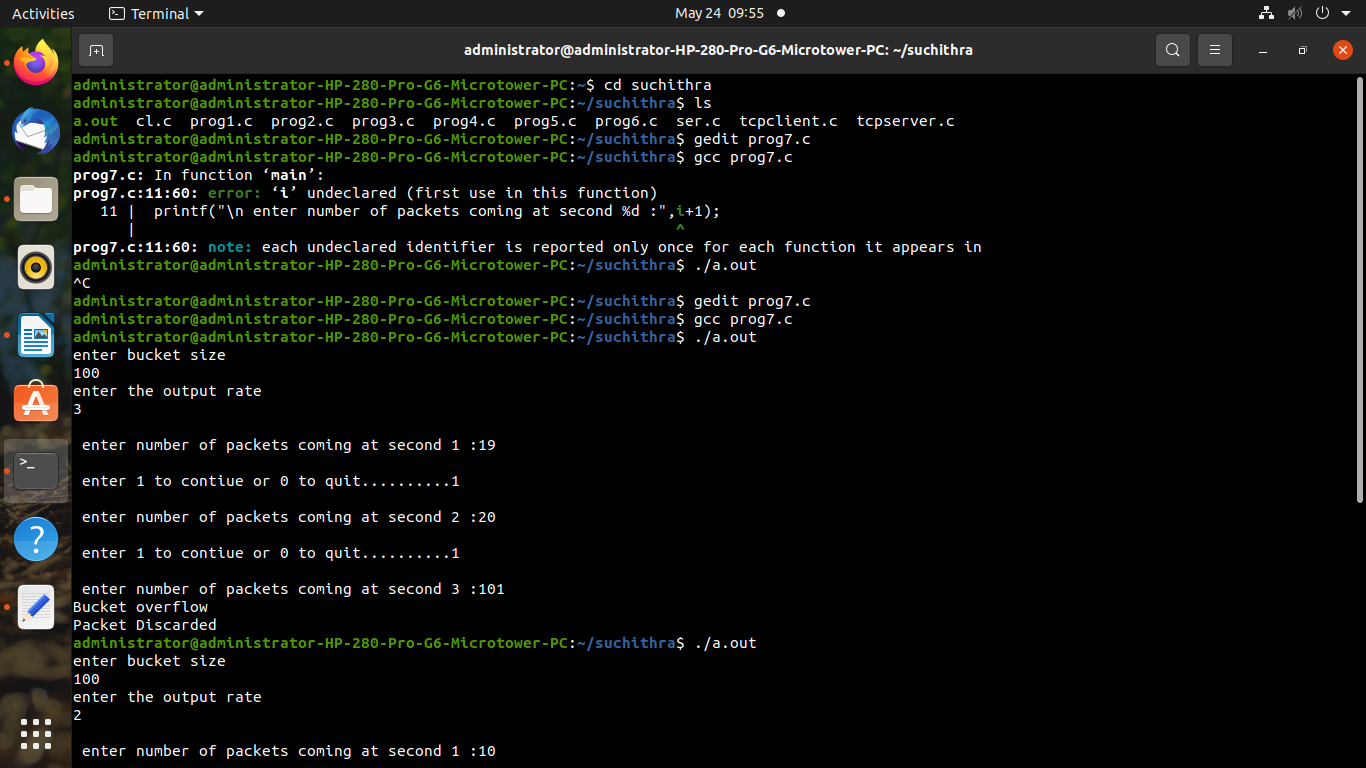
printf(" \t %d\t %d \n",drop,count);

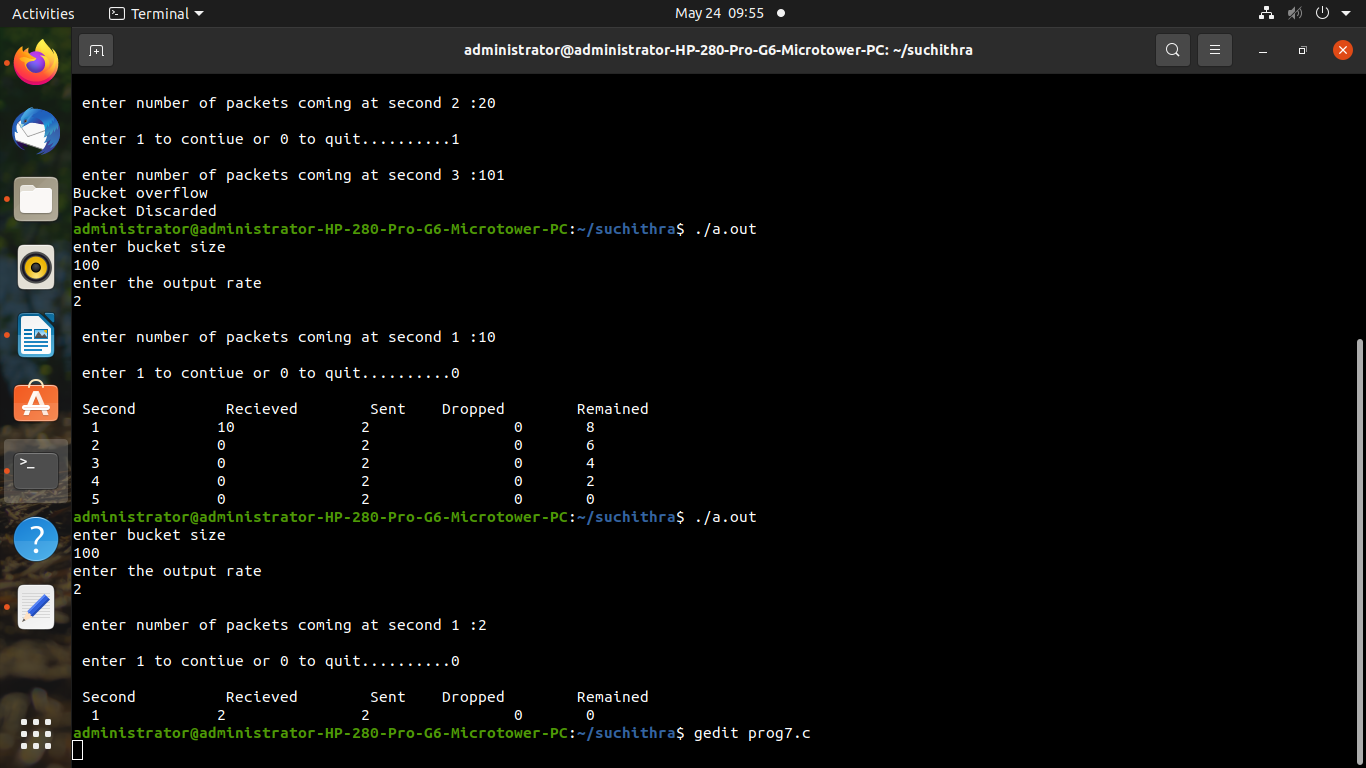
}

return 0;

}

OUTPUT:





PART B

1. create udp echo client and server application on p2p connection.

#include "ns3/netanim-module.h"

#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/internet-module.h"

#include "ns3/point-to-point-module.h"

#include "ns3/applications-module.h"

using namespace ns3;

int main(int argc,char \*argv[])

{

Time::SetResolution (Time::NS);

//creating nodes

NodeContainer nodes;

nodes.Create(2);

//create link

PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute("DataRate",StringValue("5Mbps"));

pointToPoint.SetChannelAttribute("Delay",StringValue("2ms"));

//join link to node

NetDeviceContainer devices;

devices=pointToPoint.Install (nodes);

//install protocol stack

InternetStackHelper stack;

stack.Install(nodes);

//set base address

Ipv4AddressHelper address;

address.SetBase("10.1.1.0","255.255.255.0");

//assign address

Ipv4InterfaceContainer interfaces=address.Assign(devices);

//application start server at port 9

UdpEchoServerHelper echoServer(9);

//install server on node 1

ApplicationContainer serverApps=echoServer.Install(nodes.Get(1));

//start and stop the server

serverApps.Start(Seconds(1.0));

serverApps.Stop(Seconds(10.0));

//server address and port number

UdpEchoClientHelper echoClient(interfaces.GetAddress(1),9);

echoClient.SetAttribute("MaxPackets",UintegerValue(1));

echoClient.SetAttribute("Interval",TimeValue(Seconds(1.0)));

echoClient.SetAttribute("PacketSize",UintegerValue(1024));

ApplicationContainer clientApps=echoClient.Install(nodes.Get(0));

clientApps.Start(Seconds(2.0));

clientApps.Stop(Seconds(10.0));

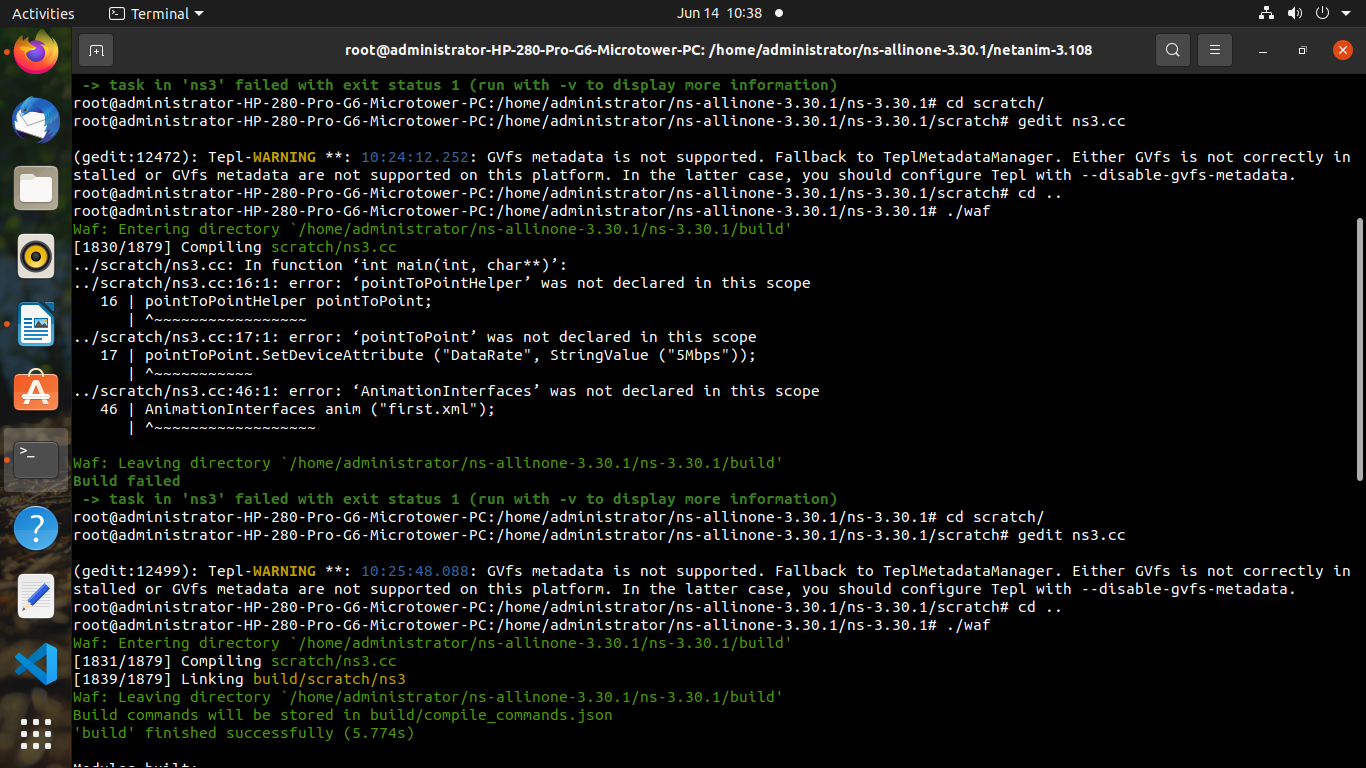
AnimationInterface anim("first.xml");

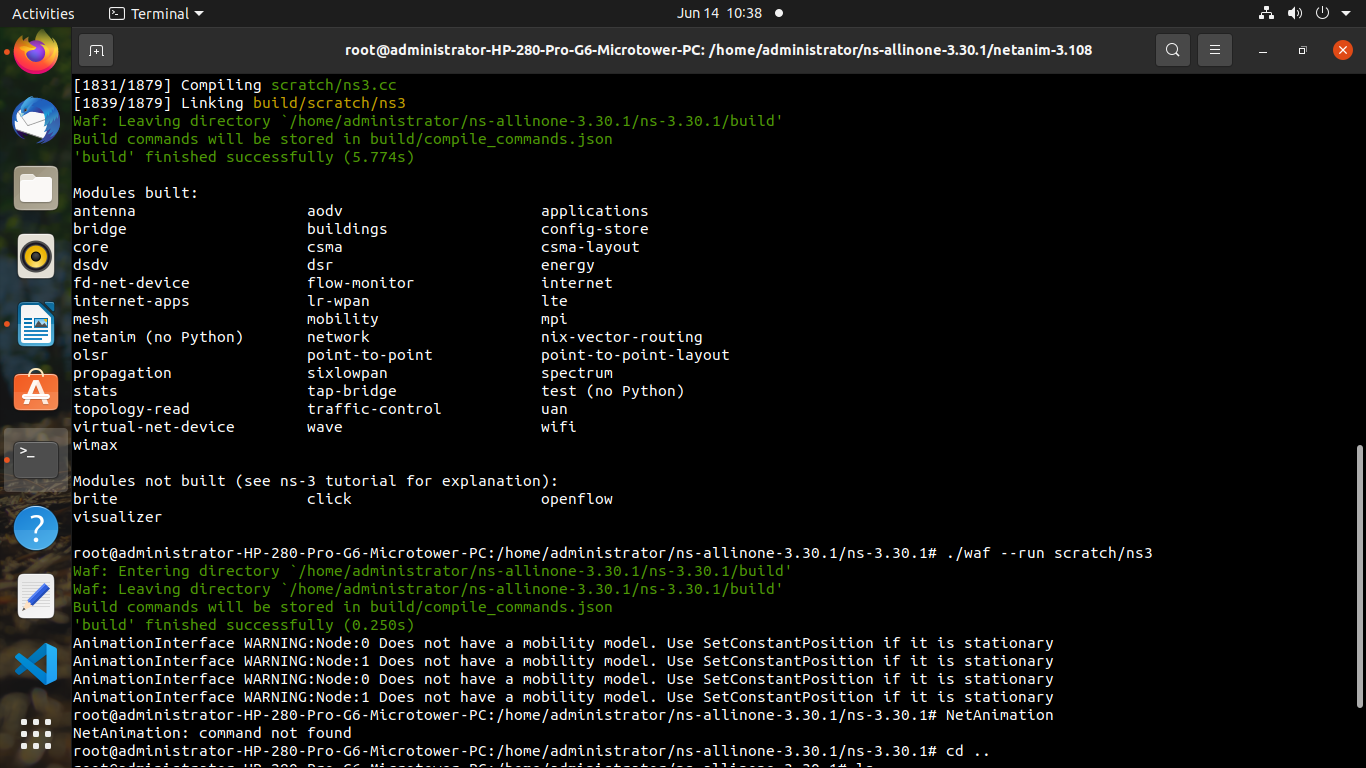
Simulator::Run();

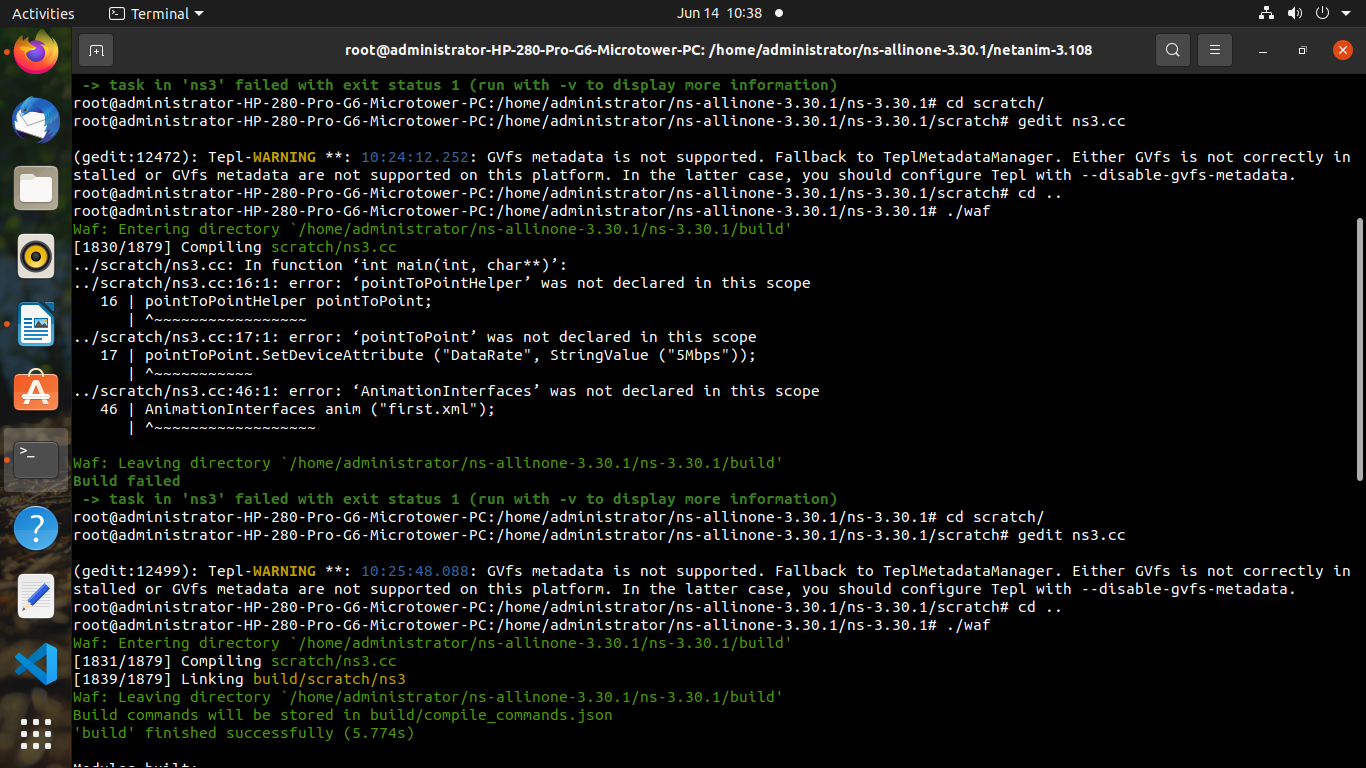
Simulator::Destroy();

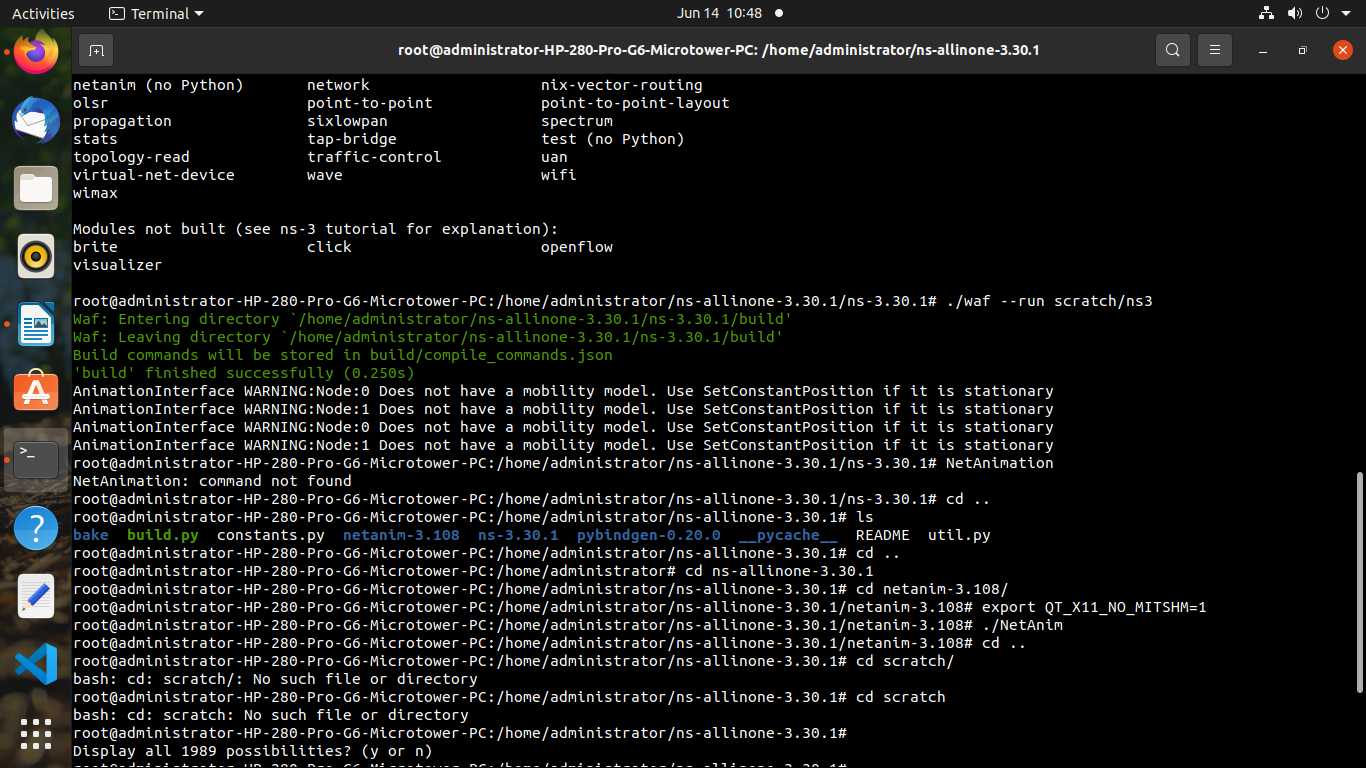
return 0;

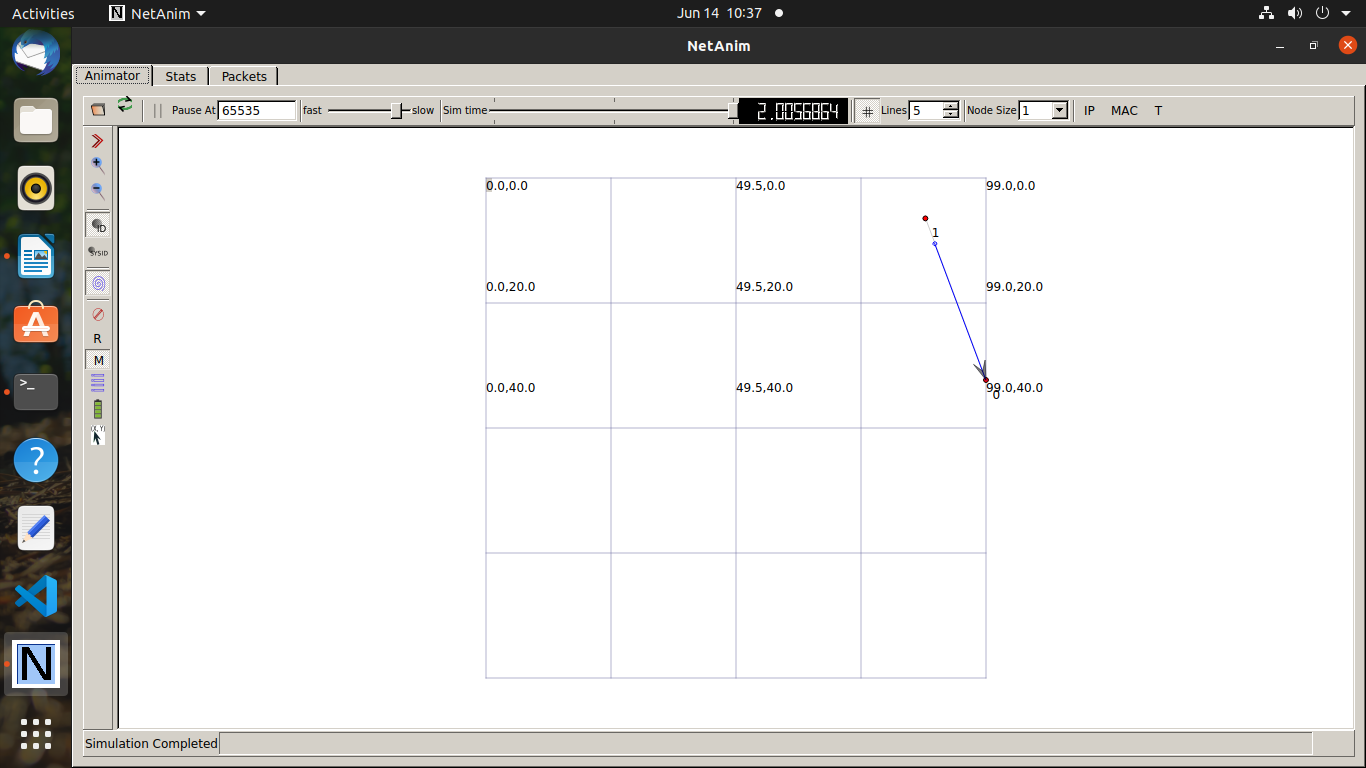
}











program 2.

#include <fstream>

#include "ns3/core-module.h"

#include "ns3/csma-module.h"

#include "ns3/applications-module.h"

#include "ns3/internet-module.h"

#include "ns3/netanim-module.h"

using namespace ns3;

int

main (int argc, char \*argv[])

{

Address serverAddress;

NodeContainer n;

n.Create (4);

InternetStackHelper internet;

internet.Install (n);

CsmaHelper csma;

csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (5000000)));

csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));

csma.SetDeviceAttribute ("Mtu", UintegerValue (1400));

NetDeviceContainer d = csma.Install (n);

Ipv4AddressHelper ipv4;

ipv4.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer i = ipv4.Assign (d);

serverAddress = Address(i.GetAddress (1));

uint16\_t port = 9; // well-known echo port number

UdpEchoServerHelper server (port);

ApplicationContainer apps = server.Install (n.Get (1));

apps.Start (Seconds (1.0));

apps.Stop (Seconds (10.0));

uint32\_t packetSize = 1024;

uint32\_t maxPacketCount = 1;

Time interPacketInterval = Seconds (1.);

UdpEchoClientHelper client (serverAddress, port);

client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount));

client.SetAttribute ("Interval", TimeValue (interPacketInterval));

client.SetAttribute ("PacketSize", UintegerValue (packetSize));

apps = client.Install (n.Get (0));

apps.Start (Seconds (2.0));

apps.Stop (Seconds (10.0));

#if 0

client.SetFill (apps.Get (0), "Hello World");

client.SetFill (apps.Get (0), 0xa5, 1024);

uint8\_t fill[] = { 0, 1, 2, 3, 4, 5, 6};

client.SetFill (apps.Get (0), fill, sizeof(fill), 1024);

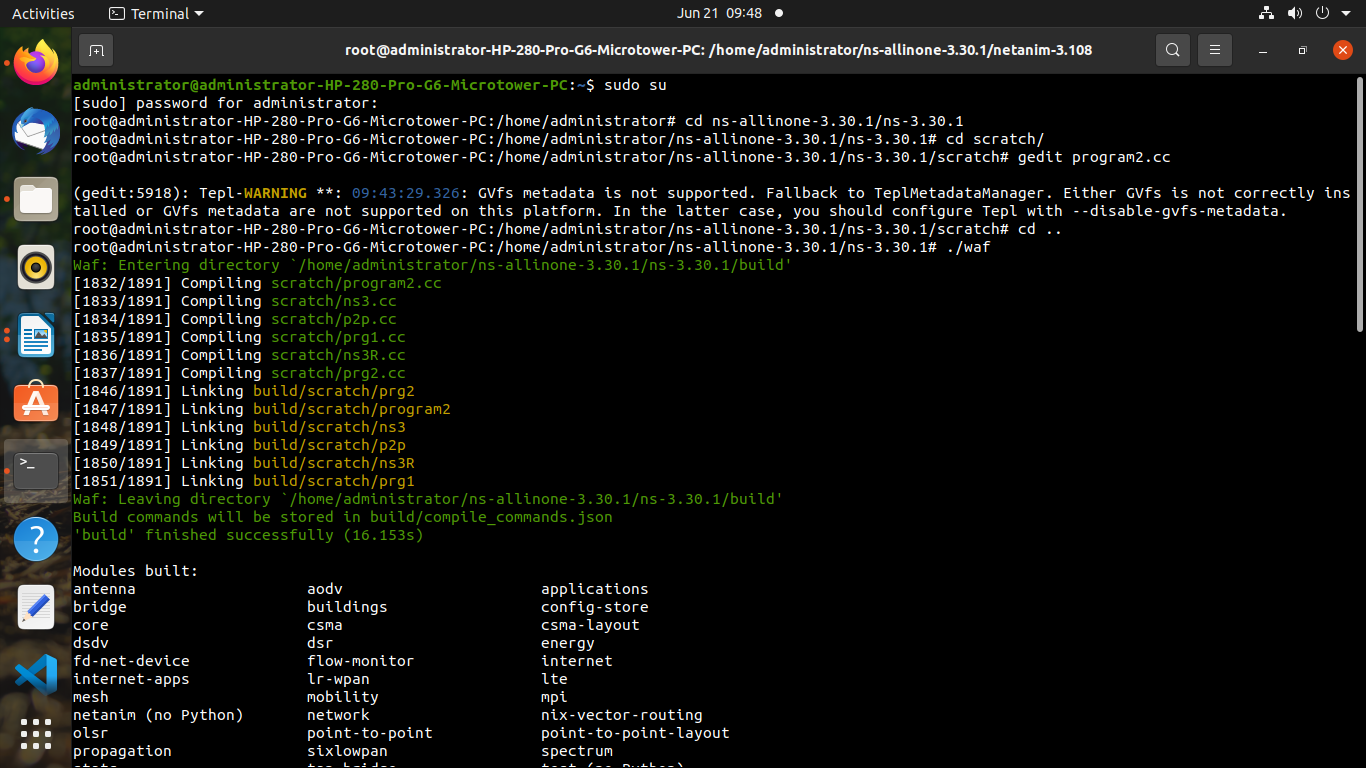
#endif

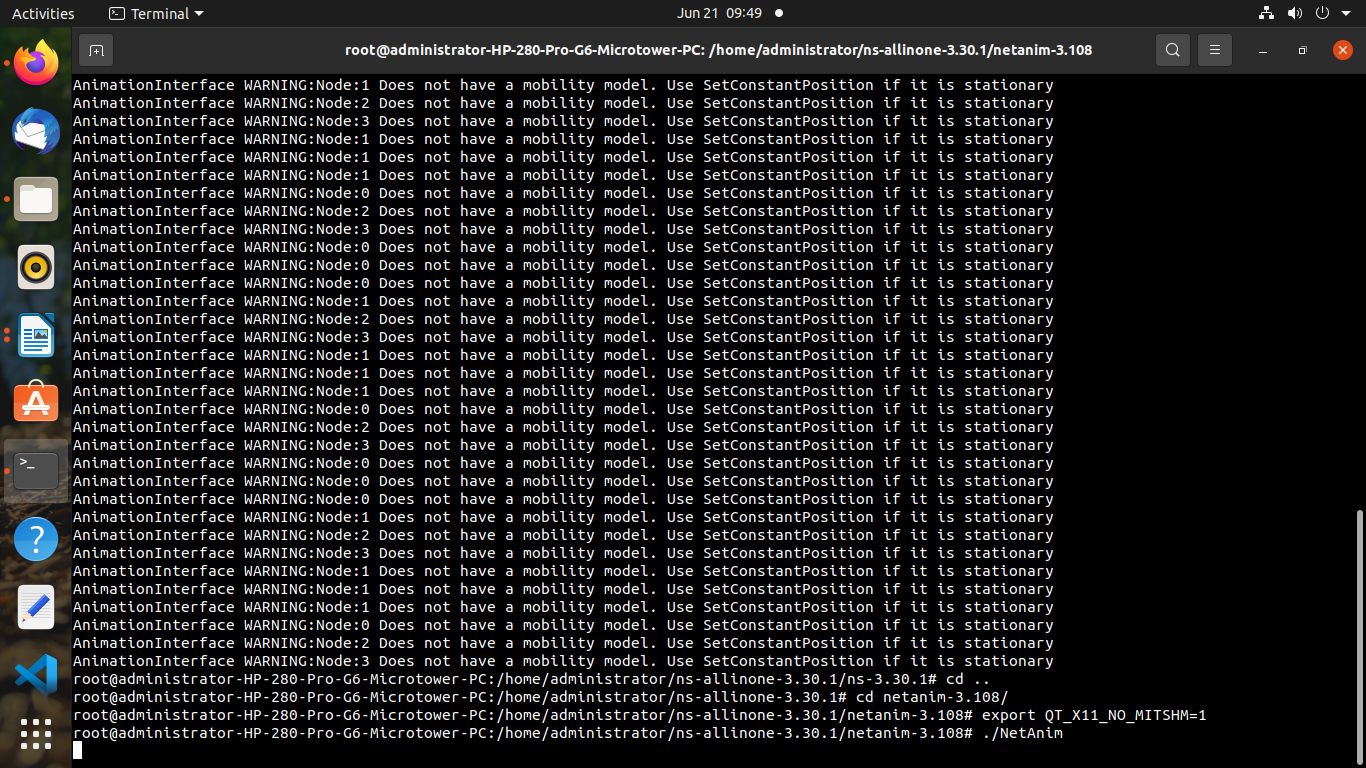
AnimationInterface anim ("second.xml");

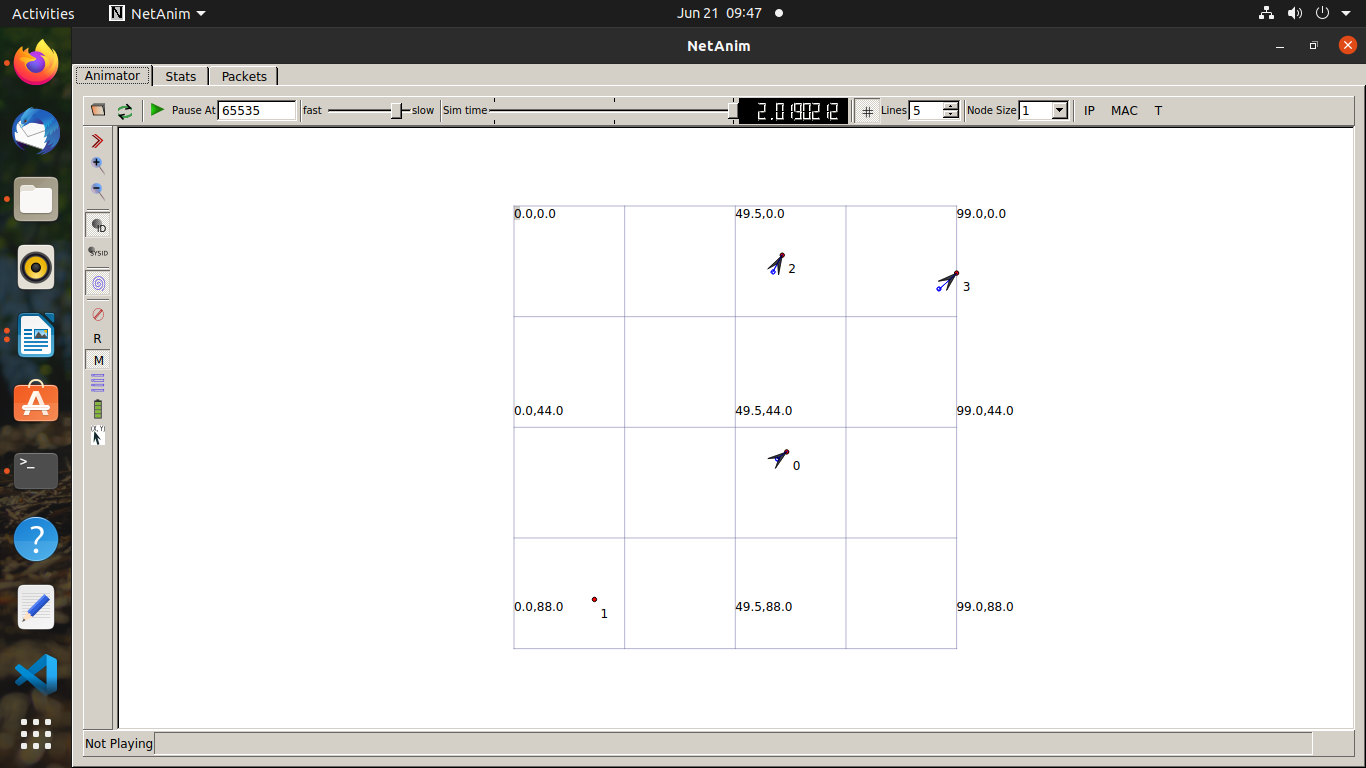
Simulator::Run ();

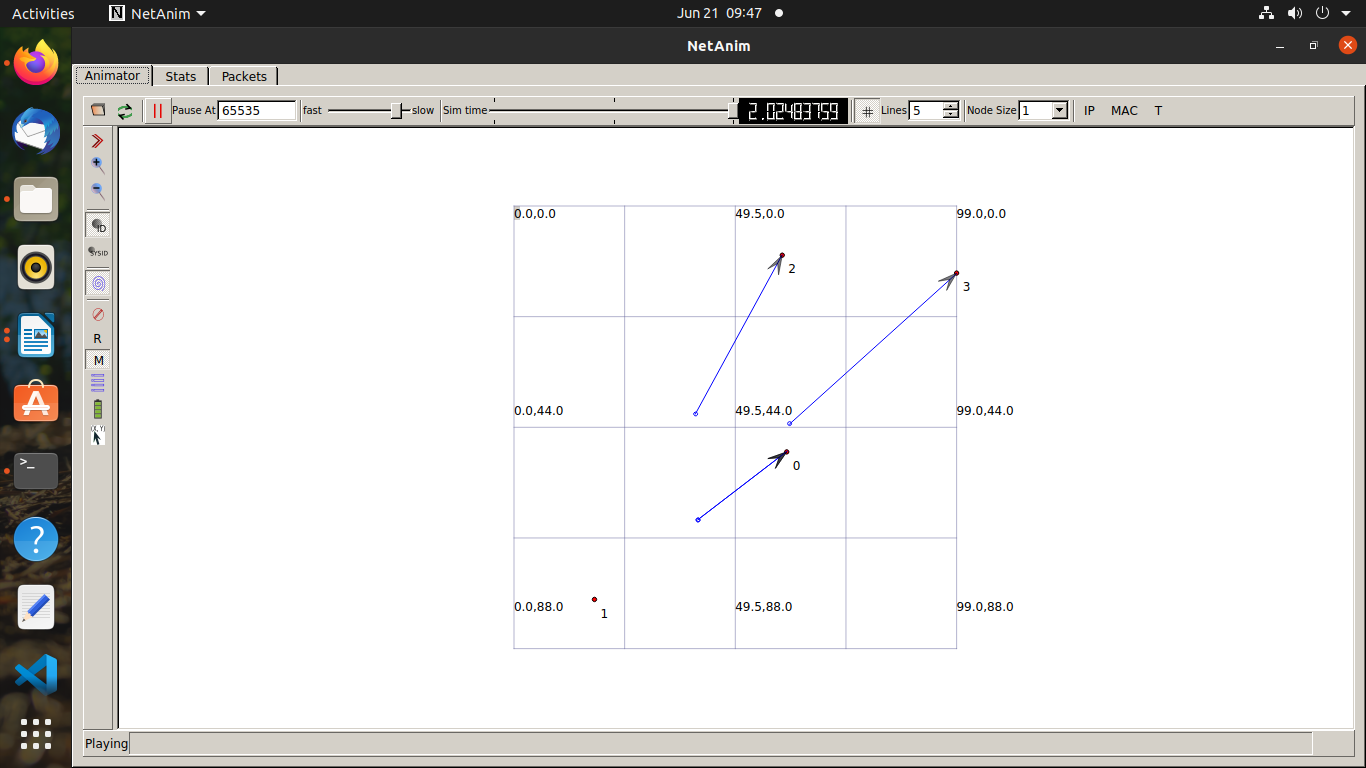
Simulator::Destroy ();

}









program3

#include "ns3/core-module.h"

#include "ns3/point-to-point-module.h"

#include "ns3/network-module.h"

#include "ns3/applications-module.h"

#include "ns3/wifi-module.h"

#include "ns3/mobility-module.h"

#include "ns3/csma-module.h"

#include "ns3/internet-module.h"

#include "ns3/netanim-module.h"

using namespace ns3;

int

main (int argc, char \*argv[])

{

uint32\_t nCsma = 3;

NodeContainer p2pNodes;

p2pNodes.Create (2);

NodeContainer csmaNodes;

csmaNodes.Add (p2pNodes.Get (1));

csmaNodes.Create (nCsma);

PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));

pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer p2pDevices;

p2pDevices = pointToPoint.Install (p2pNodes);

CsmaHelper csma;

csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));

csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices;

csmaDevices = csma.Install (csmaNodes);

InternetStackHelper stack;

stack.Install (p2pNodes.Get (0));

stack.Install (csmaNodes);

Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer p2pInterfaces;

p2pInterfaces = address.Assign (p2pDevices);

address.SetBase ("10.1.2.0", "255.255.255.0");

Ipv4InterfaceContainer csmaInterfaces;

csmaInterfaces = address.Assign (csmaDevices);

UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (nCsma));

serverApps.Start (Seconds (1.0));

serverApps.Stop (Seconds (10.0));

UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (nCsma), 9);

echoClient.SetAttribute ("MaxPackets", UintegerValue (1));

echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));

echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));

clientApps.Start (Seconds (2.0));

clientApps.Stop (Seconds (10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

pointToPoint.EnablePcapAll ("second");

csma.EnablePcap ("second", csmaDevices.Get (1), true);

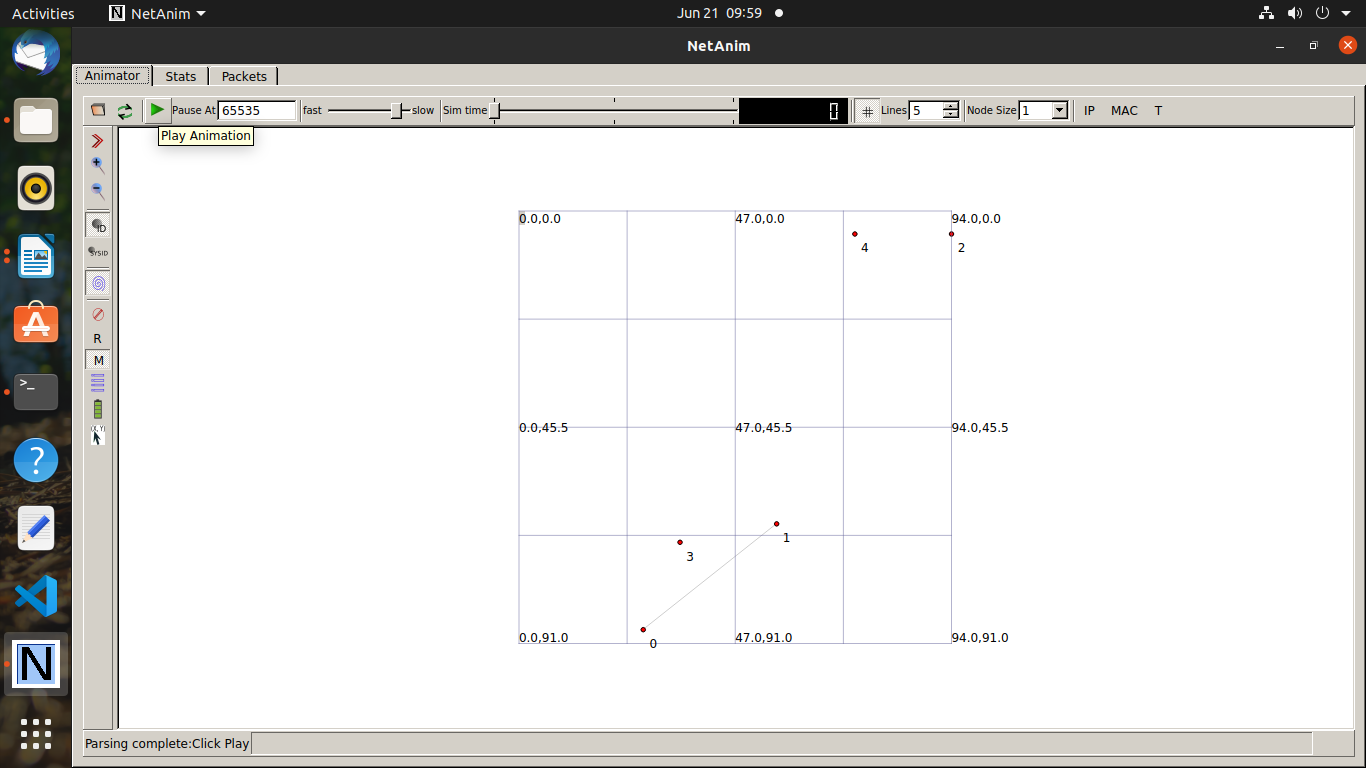
AnimationInterface anim ("third.xml");

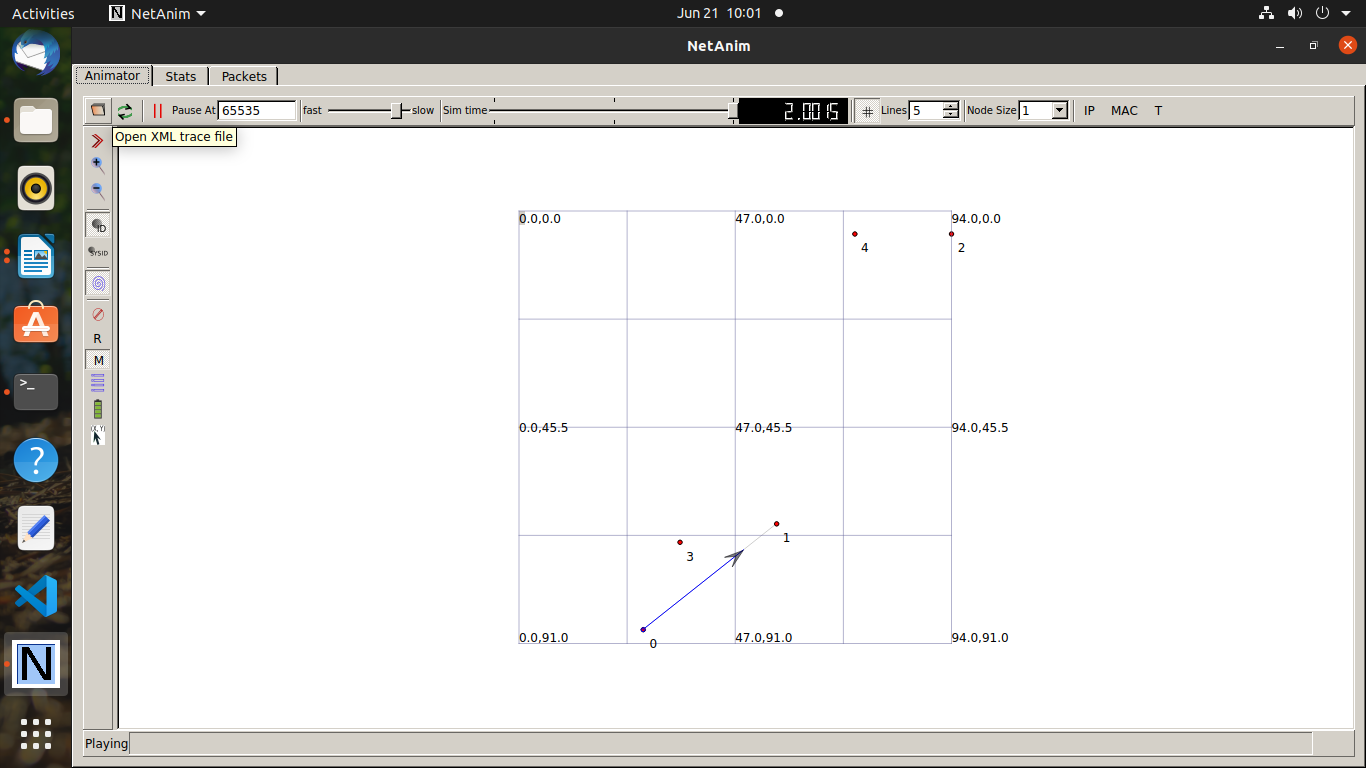
Simulator::Run ();

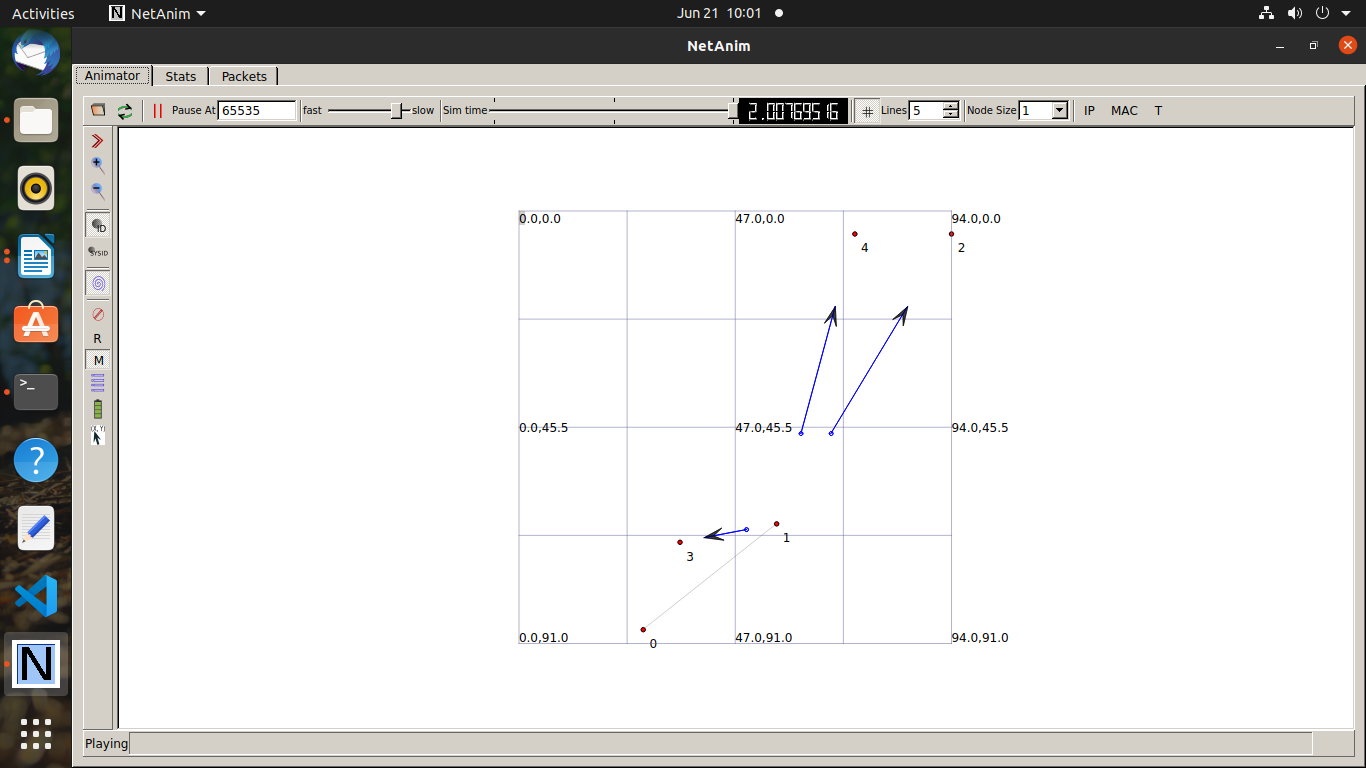
Simulator::Destroy ();

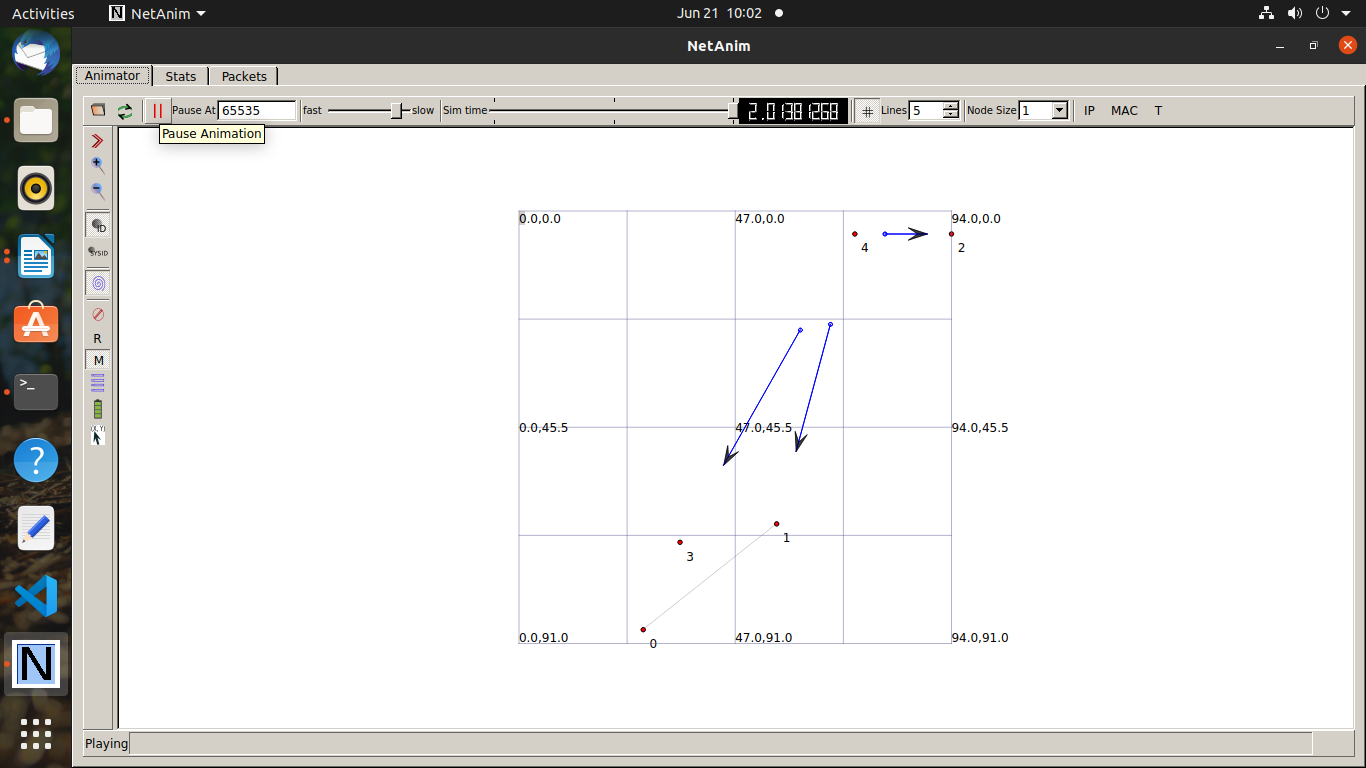
return 0;

}









program4

#include <string>

#include <fstream>

#include "ns3/core-module.h"

#include "ns3/point-to-point-module.h"

#include "ns3/internet-module.h"

#include "ns3/applications-module.h"

#include "ns3/network-module.h"

#include "ns3/packet-sink.h"

#include "ns3/netanim-module.h"

using namespace ns3;

int

main (int argc, char \*argv[])

{

uint32\_t maxBytes = 0;

NodeContainer nodes;

nodes.Create (2);

PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("500Kbps"));

pointToPoint.SetChannelAttribute ("Delay", StringValue ("5ms"));

NetDeviceContainer devices;

devices = pointToPoint.Install (nodes);

InternetStackHelper internet;

internet.Install (nodes);

Ipv4AddressHelper ipv4;

ipv4.SetBase ("10.1.1.0", "255.255.255.0");

Ipv4InterfaceContainer i = ipv4.Assign (devices);

uint16\_t port = 9; // well-known echo port number

BulkSendHelper source ("ns3::TcpSocketFactory",

InetSocketAddress (i.GetAddress (1), port));

source.SetAttribute ("MaxBytes", UintegerValue (maxBytes));

ApplicationContainer sourceApps = source.Install (nodes.Get (0));

sourceApps.Start (Seconds (0.0));

sourceApps.Stop (Seconds (10.0));

PacketSinkHelper sink ("ns3::TcpSocketFactory",

InetSocketAddress (Ipv4Address::GetAny (), port));

ApplicationContainer sinkApps = sink.Install (nodes.Get (1));

sinkApps.Start (Seconds (0.0));

sinkApps.Stop (Seconds (10.0));

Simulator::Stop (Seconds (10.0));

AnimationInterface anim ("fourth.xml");

anim.EnablePacketMetadata(true);

Simulator::Run ();

Simulator::Destroy ();

}

