Roll No:-C22078
Name:-Pratik Vilas Mestry

Deccan Education Society's

Navinchandra Mehta Institute of Technology and Development

C E R T I F I C A T E

This is to certify that Mr. / ~~Miss.~~ **Pratik Vilas Mestry** of M.C. A. Semester II with Roll No. **C22078** has completed 14 practicals of MCAL26 Networking with Linux under my supervision in this college during the year 2022 -2023.

CO	R1: Journal	R2: Performance during lab session	R3: Implementation using different problem-solving techniques	R4: Mock Viva	Attendance
CO1					
CO2					
CO3					
CO4					

Practical-in-charge

Head of Department
MCA Department
(NMITD)

Roll No:-C22078
Name:-Pratik Vilas Mestry

Index				
Sr. No	**Problem Statement**	**Course Outcome**	**Date**	**Sign**
1	Installation of NS-3 in Linux	CO1		
2	Installation of NS-3 in Linux	CO1		
3	Installation of NS-3 in Linux	CO1		
4	Program to simulate traffic between two nodes	CO2		
5	Program to simulate star topology	CO2		
6	Program to simulate bus topology	CO2		
7	Program to simulate mesh topology	CO2		
8	Program to simulate hybrid topology	CO2		
9	Program to simulate UDP server	CO2, CO3		
10	Program to simulate UDP server client	CO2, CO3		
11	Program to simulate DHCP server and n clients	CO2, CO3		
12	Animate a simple network using Net Anim in Network Simulator	CO3		
13	Analyze the network traffic using Wire Shark	CO4		
14	Mini Project:- Animate 3 way hanshake for TCP connection using NetAnimator	CO4		

Roll No:-C22078
Name:-Pratik Vilas Mestry

Practical No. 1

Aim: Installing NS-3 in Ubuntu

Steps for installing NS-3 in Ubuntu

Step 1: Update the system

$ sudo apt update

Step 2: Prerequisites for installing NS-3

$ sudo apt install build-essential autoconf automake libxmu-dev g++ python3 python3-dev pkg-config sqlite3 cmake python3-setuptools git qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools gir1.2-goocanvas-2.0 python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython3 openmpi-bin openmpi-common openmpi-doc libopenmpi-dev autoconf cvs bzr unrar gsl-bin libgsl-dev libgslcblas0 wireshark tcpdump sqlite sqlite3 libsqlite3-dev libxml2 libxml2-dev libc6-dev libc6-dev-i386 libclang-dev llvm-dev automake python3-pip libxml2 libxml2-dev libboost-all-dev

Now download the ns3 3.35 from https://nsnam.org

Copy the softwares from the Downloads/ folder to the home folder (in my case its /home/ns-3/)

Now extract both the versions using the GUI method.

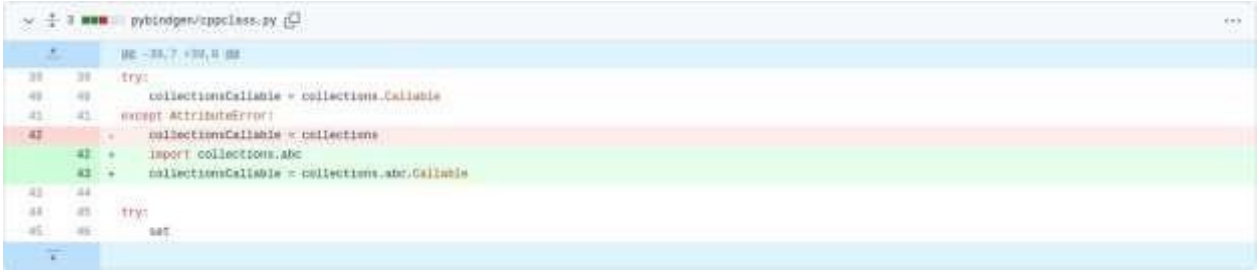
Just right click and click "Extract Here"

Now we will install ns-3.35

$ cd
$ cd ns-allinone-3.35/
$ ./build.py --enable-examples --enable-tests

In case, if you get the following error pybindgen(ns3 module antenna)

Do this step and repeat the above step



We have installed two version of ns-3.35 successfully in Ubuntu.

Roll No:-C22078
Name:-Pratik Vilas Mestry

Practical No. 2

Aim: Install NetAnim in Ubuntu

Steps to Install NetAnim

You can directly install NetAnim

Otherwise, you have to execute some commands but for this we need NS3installed or compiled.

Step1: sudo apt-get install NetAnim

Step2: NetAnim file.xml

Step3: Select Xml File

Step4: Run the simulation by clicking, NS3 NetAnim successfully

Roll No:-C22078
Name:-Pratik Vilas Mestry

Practical No. 3

Aim: Install Wireshark In Ubuntu

Install Wireshark

Step 1: Add the stable official PPA. To do this, go to terminal by pressing Ctrl+Alt+T and run:

sudo add-apt-repository ppa:wireshark-dev/stable

Step 2: Update the repository:

sudo apt-get update

Step 3: Install wireshark 2.0:

sudo apt-get install wireshark

Step 4: Run wireshark:

sudo wireshark

If you get a error couldn't run /usr/bin/dumpcap in child process: Permission Denied. go to the terminal again and run:

sudo dpkg-reconfigure wireshark-common

Say YES to the message box. This adds a wireshark group. Then add user to the group by typing

sudo adduser $USER wireshark

Roll No:-C22078
Name:-Pratik Vilas Mestry

Practical No. 4

Aim: Program to simulate traffic between two nodes.

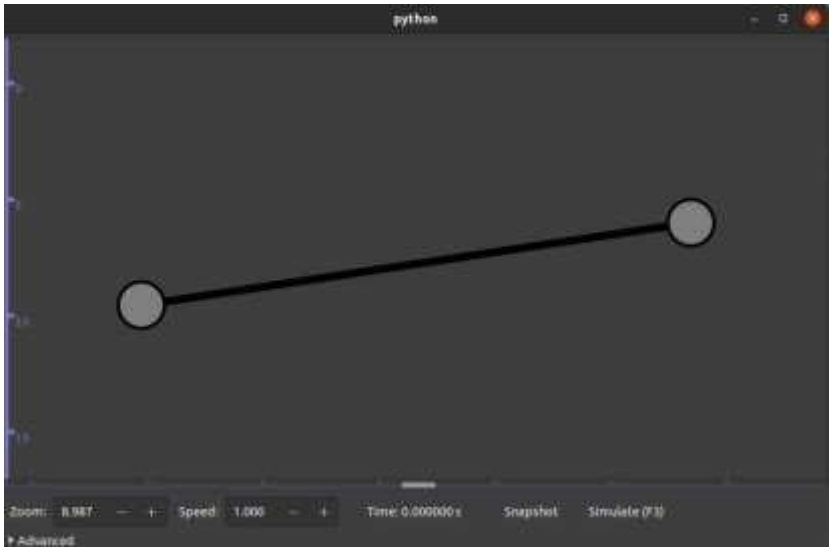
Code:

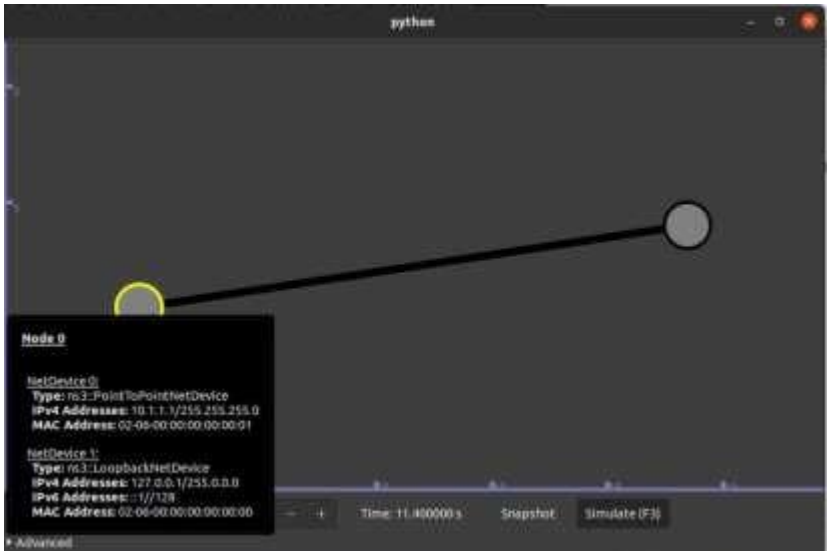
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"

//netanimator
#include "ns3/netanim-module.h"
#include "ns3/mobility-module.h"

// Default Network Topology
//
//       10.1.1.0
// n0.....n1
//    point-to-point
//

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

int
main (int argc, char *argv[])
{
  CommandLine cmd (_FILE_);
  cmd.Parse (argc, argv);

  Time::SetResolution (Time::NS);
  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

  NodeContainer nodes;
  nodes.Create (2);

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);
```

```
  InternetStackHelper stack;
  stack.Install (nodes);

  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");

  Ipv4InterfaceContainer interfaces = address.Assign (devices);

  UdpEchoServerHelper echoServer (9);

  ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1060));
  ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  MobilityHelper mobility;
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(nodes);
  AnimationInterface anim("first.xml");
  AnimationInterface::SetConstantPosition(nodes.Get(0),10,25);
  AnimationInterface::SetConstantPosition(nodes.Get(1),40,25);

  anim.EnablePacketMetadata(true);

  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

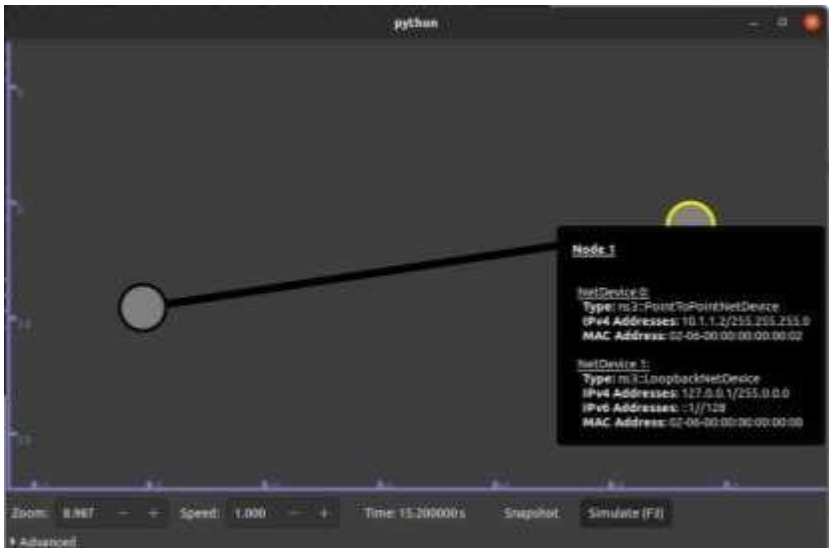
Roll No:-C22078
Name:-Pratik Vilas Mestry

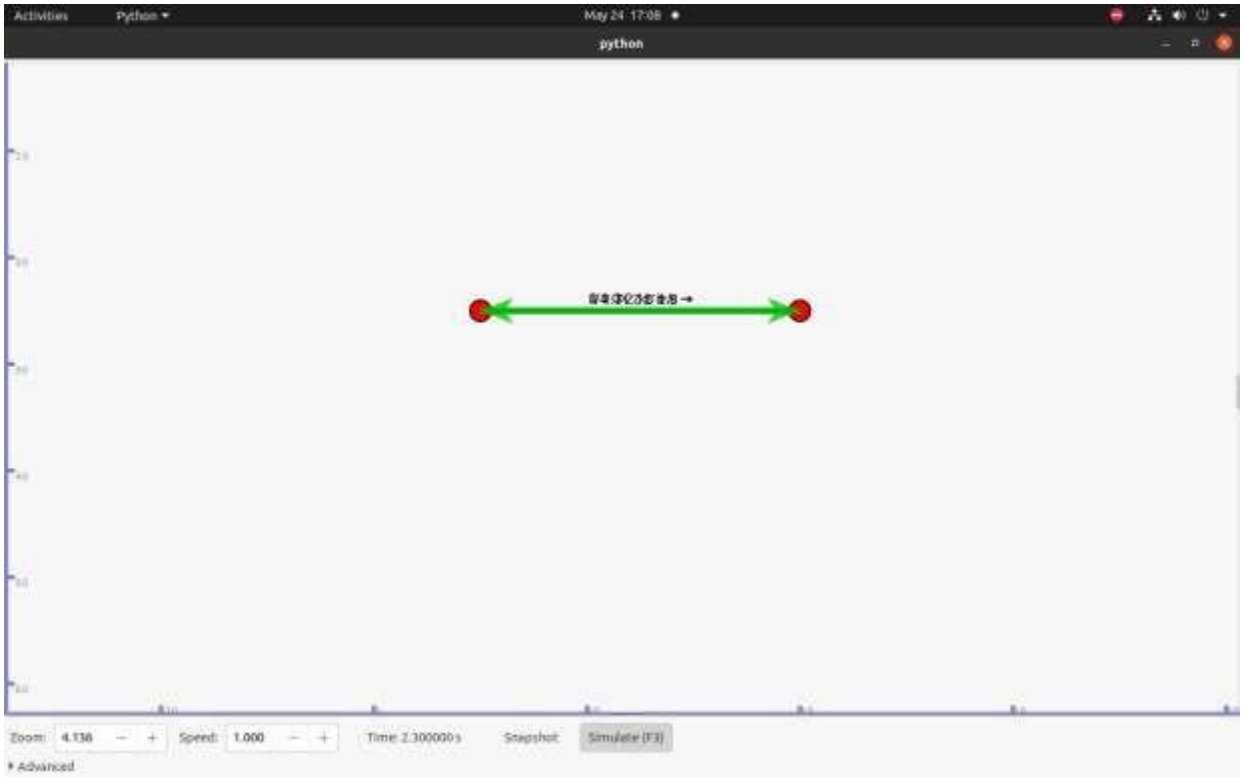
Output:



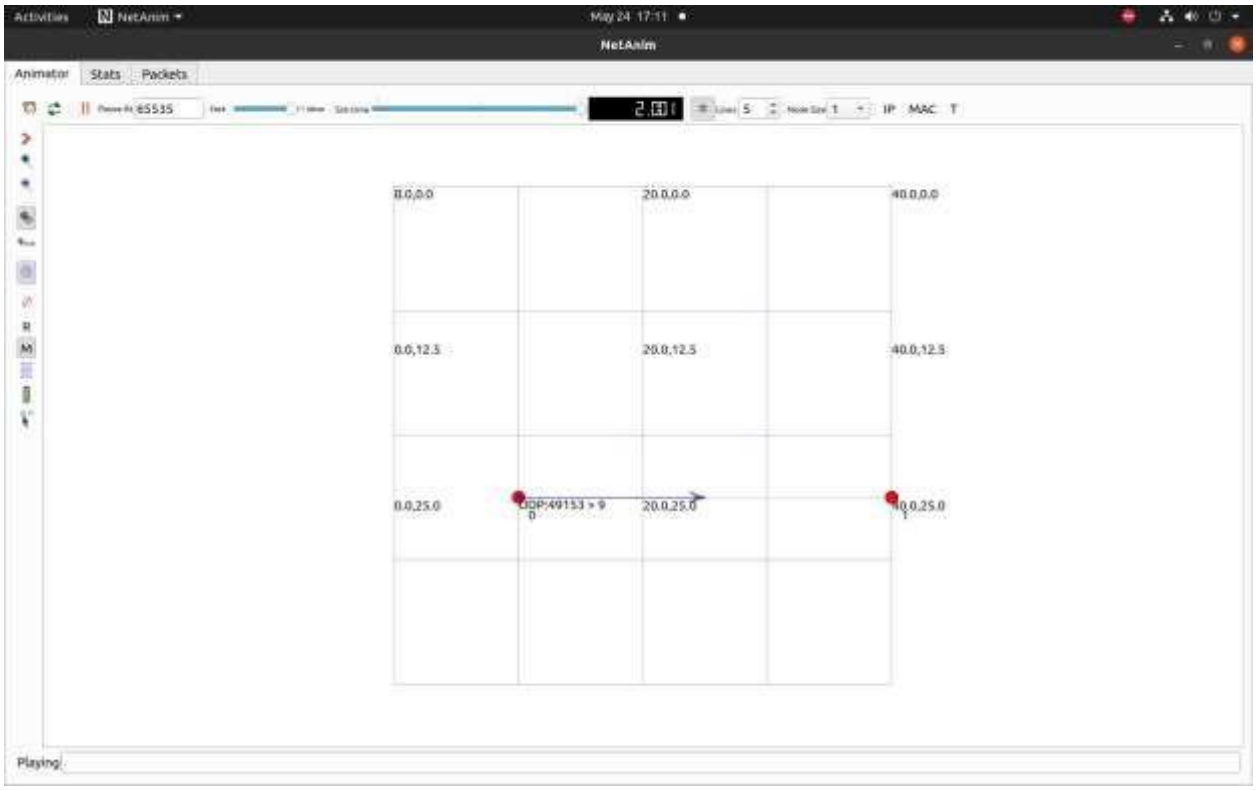


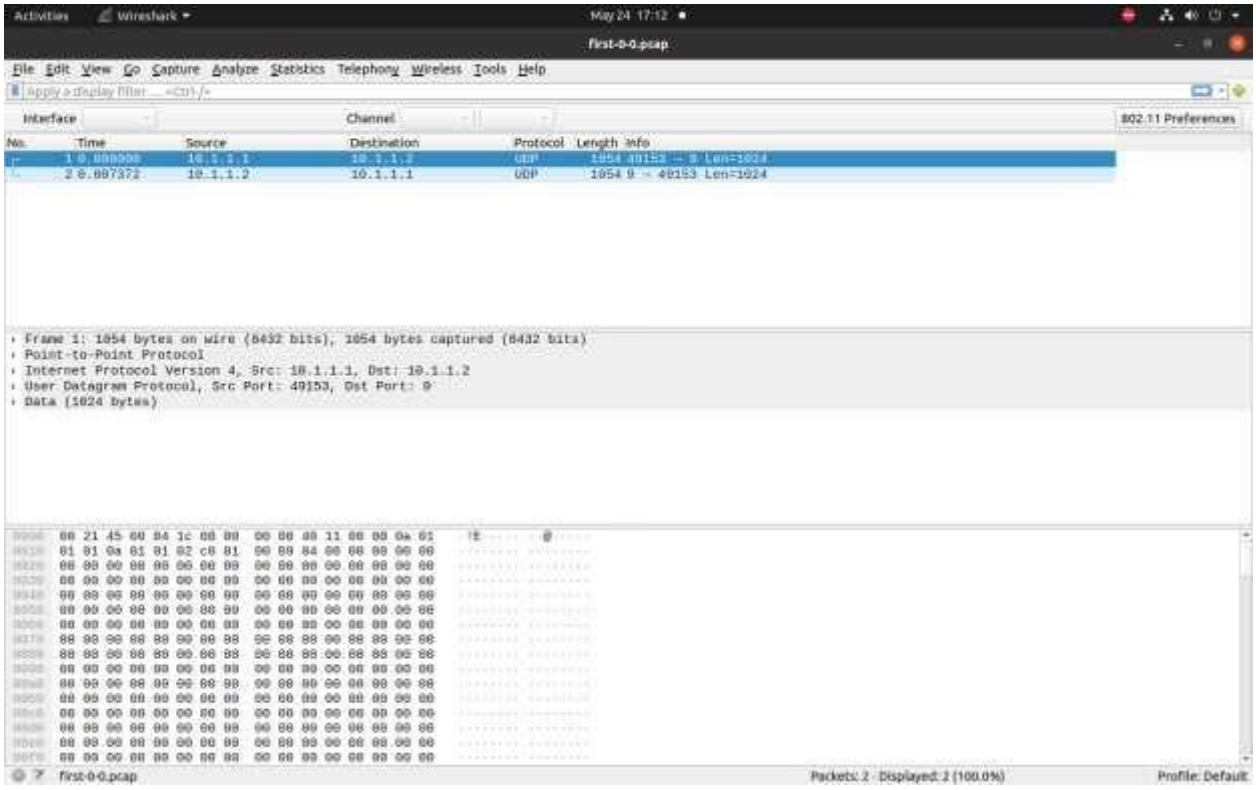
Roll No:-C22078
Name:-Pratik Vilas Mestry





Roll No:-C22078
Name:-Pratik Vilas Mestry





Roll No:-C22078
Name:-Pratik Vilas Mestry

Practical No. 5

Aim: Program to simulate star topology.

Code:

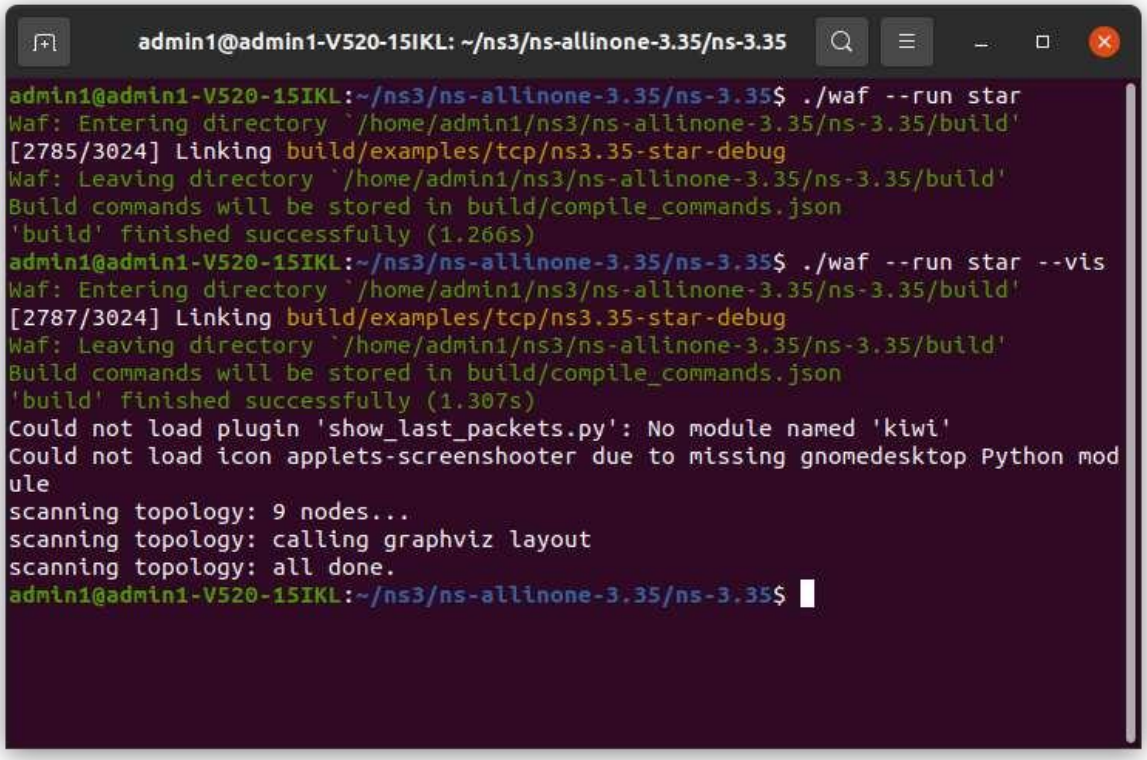
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/netanim-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/point-to-point-layout-module.h"
#include "ns3/netanim-module.h"
#include "ns3/mobility-module.h"

// Network topology (default)
//
// n2 n3 n4 .
// \ | / .
// \|/ .
// n1--- n0---n5 .
// /|\ .
// / | \ .
// n8 n7 n6 .


//
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("Star");
int main (int argc, char *argv[])
{
NodeContainer nodes;
nodes.Create(9);
//
// Set up some default values for the simulation.
//
Config::SetDefault ("ns3::OnOffApplication::PacketSize", UintegerValue
(137));
// ??? try and stick 15kb/s into the data rate
Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue
("14kb/s"));
//
// Default number of nodes in the star. Overridable by command line
argument.
//
```

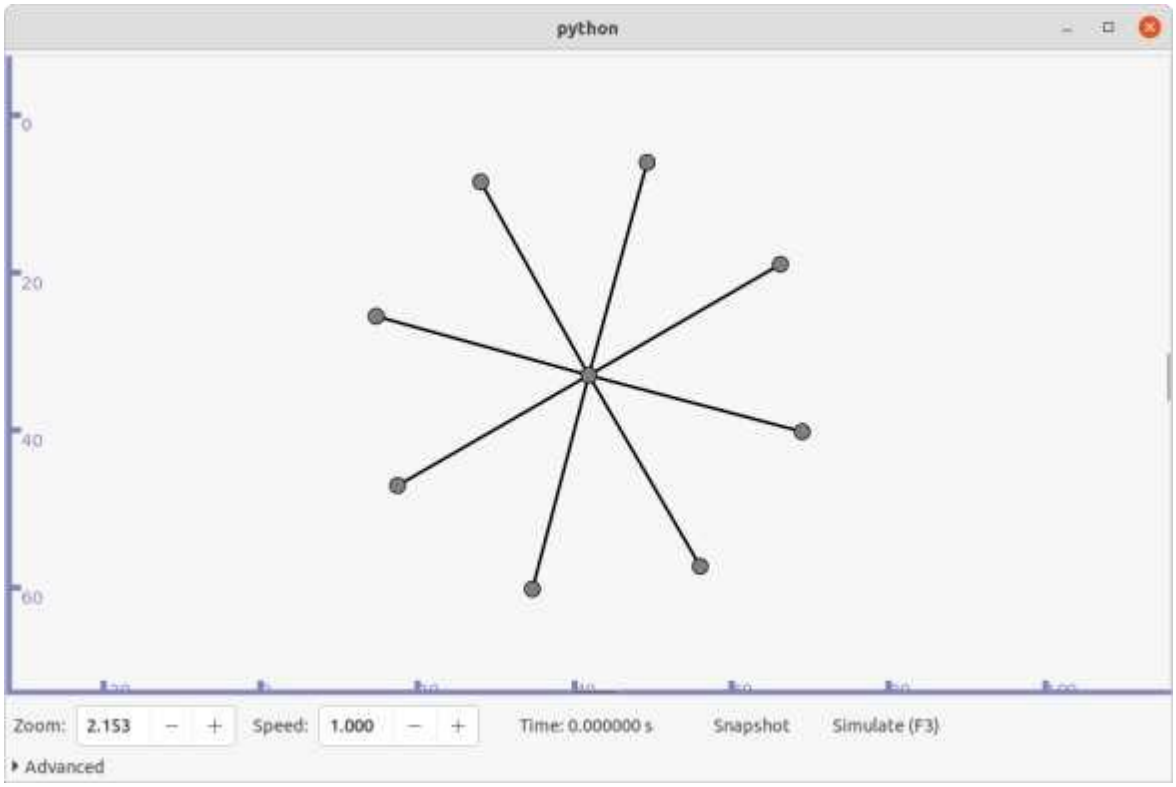
```
uint32_t nSpokes = 8;
CommandLine cmd (_FILE_);
cmd.AddValue ("nSpokes", "Number of nodes to place in the star",
nSpokes);
cmd.Parse (argc, argv);
NS_LOG_INFO ("Build star topology.");
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));


pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
PointToPointStarHelper star (nSpokes, pointToPoint);
NS_LOG_INFO ("Install internet stack on all nodes.");
InternetStackHelper internet;
star.InstallStack (internet);
NS_LOG_INFO ("Assign IP Addresses.");
star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0",
"255.255.255.0"));
NS_LOG_INFO ("Create applications.");
//
// Create a packet sink on the star "hub" to receive packets.
//
uint16_t port = 50000;
Address hubLocalAddress (InetSocketAddress (Ipv4Address::GetAny (),
port));
PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory",
hubLocalAddress);
ApplicationContainer hubApp = packetSinkHelper.Install (star.GetHub ());
hubApp.Start (Seconds (1.0));
hubApp.Stop (Seconds (10.0));
//
// Create OnOff applications to send TCP to the hub, one on each spoke
node.
//
OnOffHelper onOffHelper ("ns3::TcpSocketFactory", Address ());
onOffHelper.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));


onOffHelper.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));
ApplicationContainer spokeApps;
for (uint32_t i = 0; i < star.SpokeCount (); ++i)
```

```
{
AddressValue remoteAddress (InetSocketAddress
(star.GetHubIpv4Address (i), port));
onOffHelper.SetAttribute ("Remote", remoteAddress);
spokeApps.Add (onOffHelper.Install (star.GetSpokeNode (i)));
}
spokeApps.Start (Seconds (1.0));
spokeApps.Stop (Seconds (10.0));
NS_LOG_INFO ("Enable static global routing.");
//
// Turn on global static routing so we can actually be routed across the star.
//
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
NS_LOG_INFO ("Enable pcap tracing.");
//
// Do pcap tracing on all point-to-point devices on all nodes.
//
MobilityHelper mobility;
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(nodes);
AnimationInterface anim("star.xml");

AnimationInterface::SetConstantPosition(nodes.Get(0),10,2);
AnimationInterface::SetConstantPosition(nodes.Get(1),11,5);
AnimationInterface::SetConstantPosition(nodes.Get(2),15,2);
AnimationInterface::SetConstantPosition(nodes.Get(3),19,7);
anim.EnablePacketMetadata(true);
pointToPoint.EnablePcapAll ("star");
NS_LOG_INFO ("Run Simulation.");
Simulator::Run  ();
Simulator::Destroy ();
NS_LOG_INFO ("Done.");
return 0;
}
```

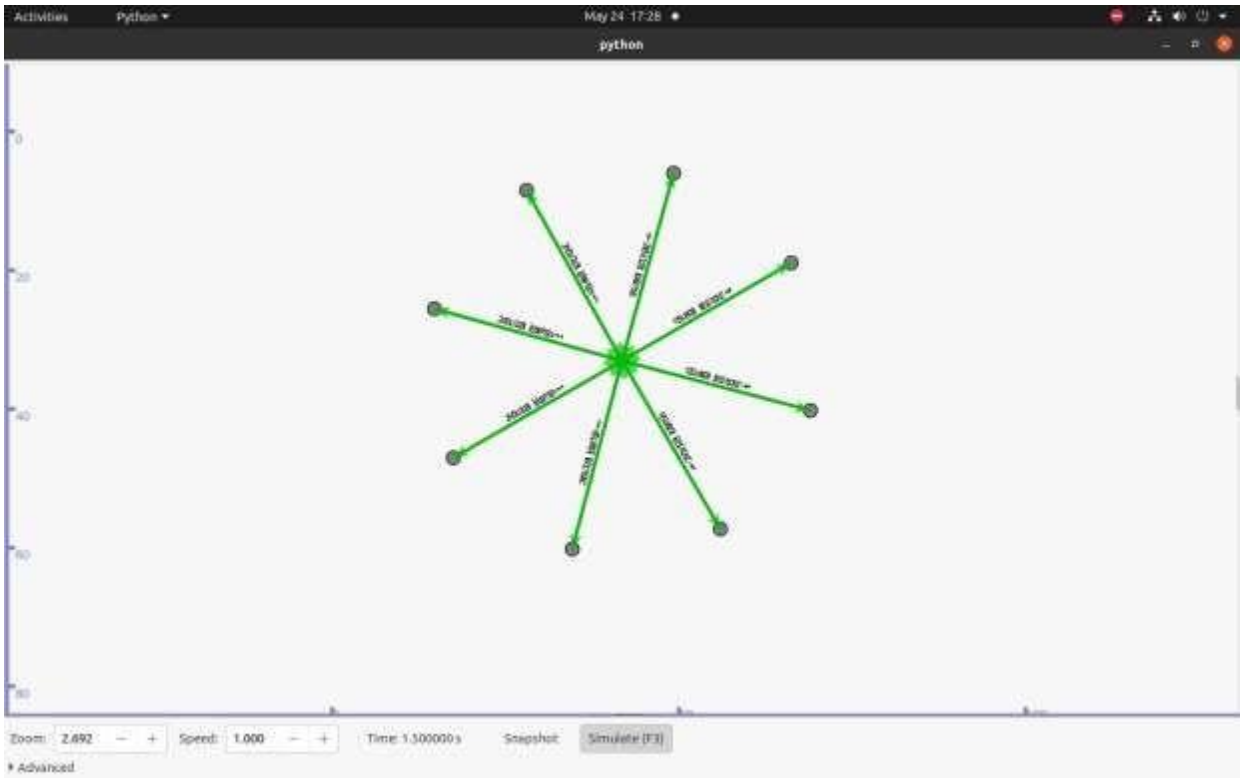
Roll No:-C22078
Name:-Pratik Vilas Mestry

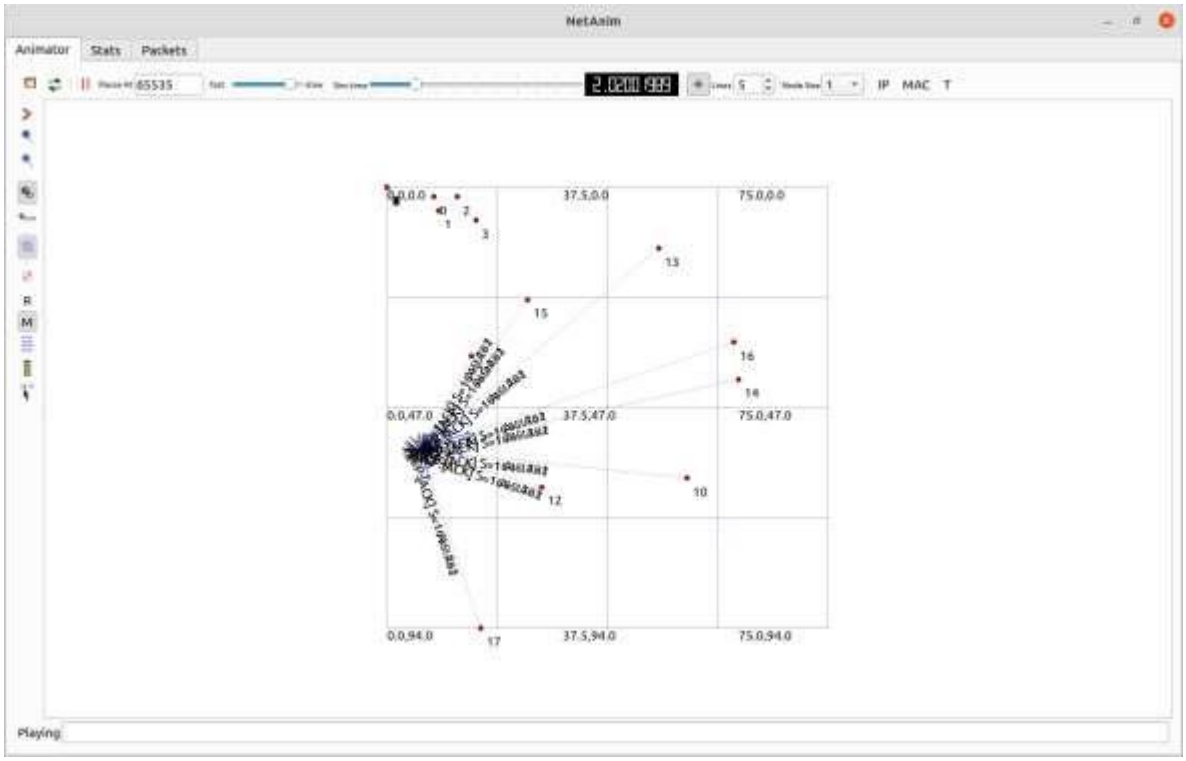
Output:



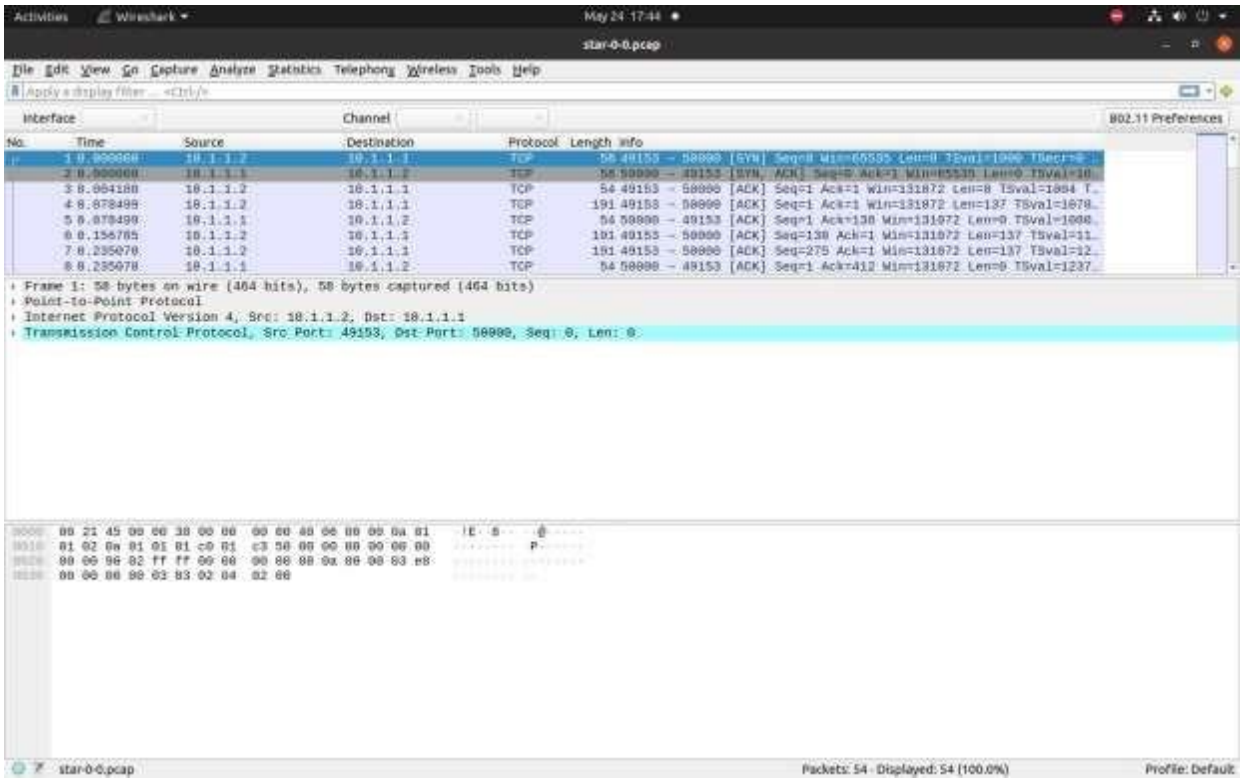


Roll No:-C22078
Name:-Pratik Vilas Mestry





Roll No:-C22078
Name:-Pratik Vilas Mestry



Roll No:-C22078
Name:-Pratik Vilas Mestry

Practical No. 6

Aim: Program to stimulate bus topology.

Code:
```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"

#include "ns3/netanim-module.h"
#include "ns3/mobility-module.h"

// Default Network Topology
//
// 10.1.1.0
// n0----------------n1 n2 n3 n4
// point-to-point | | | |
// ================
// LAN 10.1.2.0

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");
```

Roll No:-C22078
Name:-Pratik Vilas Mestry

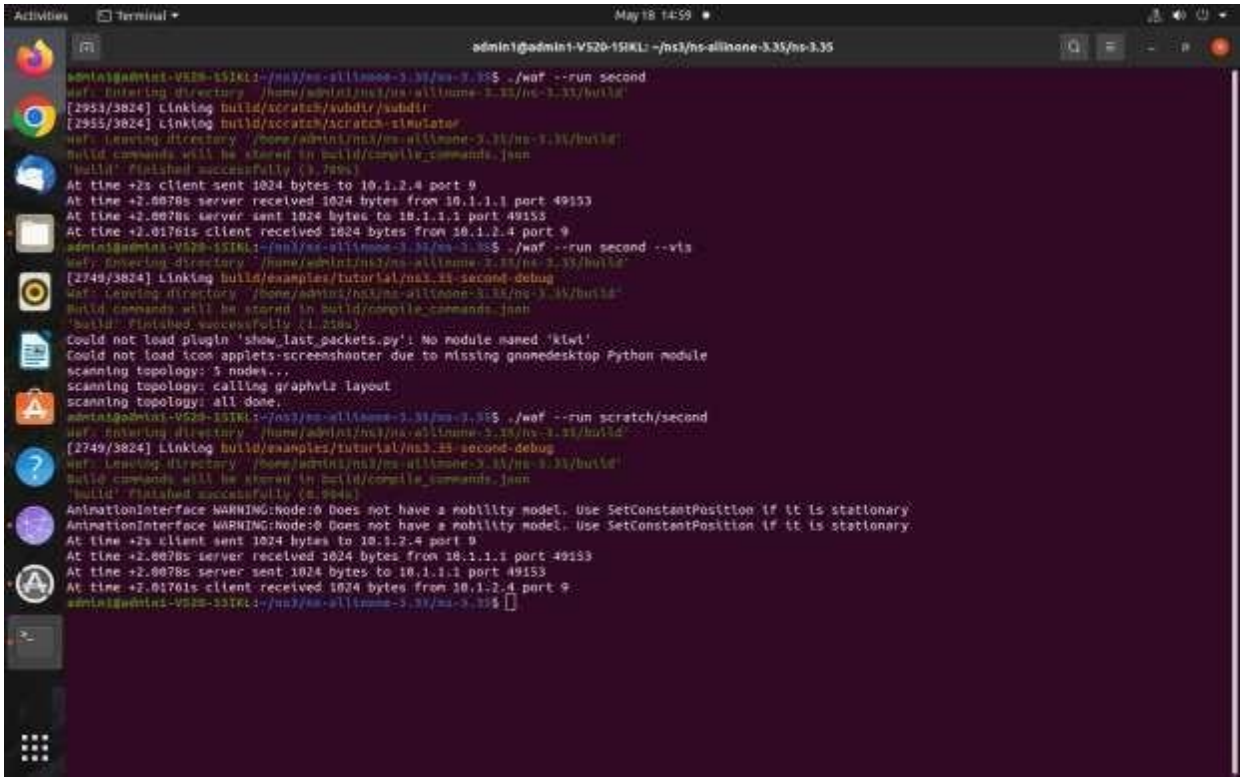
```
int
main (int argc, char *argv[])
{
bool verbose = true;
uint32_t nCsma = 3;

CommandLine cmd (__FILE__);
cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);

cmd.Parse (argc,argv);

if (verbose)
{
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

}

nCsma = nCsma == 0 ? 1 : nCsma;

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);
```

Roll No:-C22078
Name:-Pratik Vilas Mestry

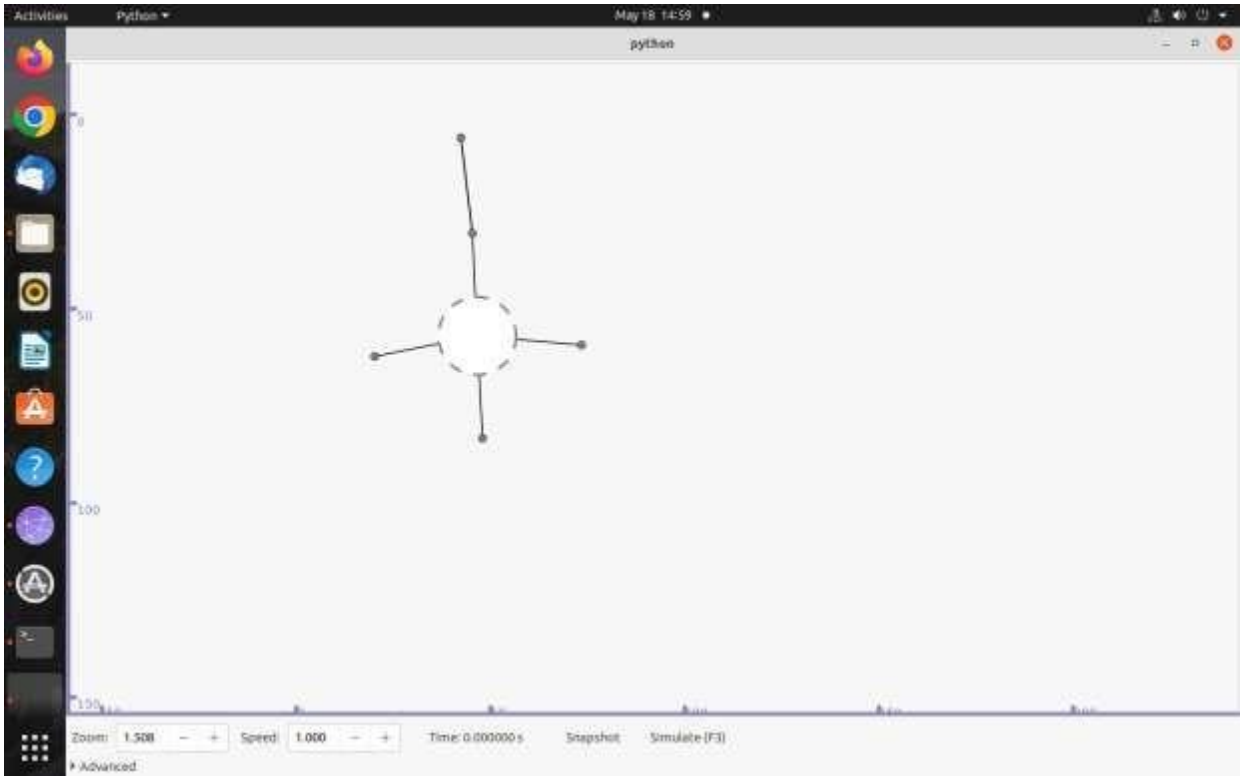
```
InternetStackHelper stack;
stack.Install (p2pNodes.Get (0));
stack.Install (csmaNodes);

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 ();

MobilityHelper mobility;
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(csmaNodes);

AnimationInterface anim("second.xml");


AnimationInterface::SetConstantPosition (csmaNodes.Get(0),10,25);
AnimationInterface::SetConstantPosition (csmaNodes.Get(1),40,25);
AnimationInterface::SetConstantPosition (csmaNodes.Get(2),30,25);
AnimationInterface::SetConstantPosition (csmaNodes.Get(3),60,25);
anim.EnablePacketMetadata(true);
```

```
pointToPoint.EnablePcapAll("second");

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



Simulator::Run ();
Simulator::Destroy ();
return 0;
}
```

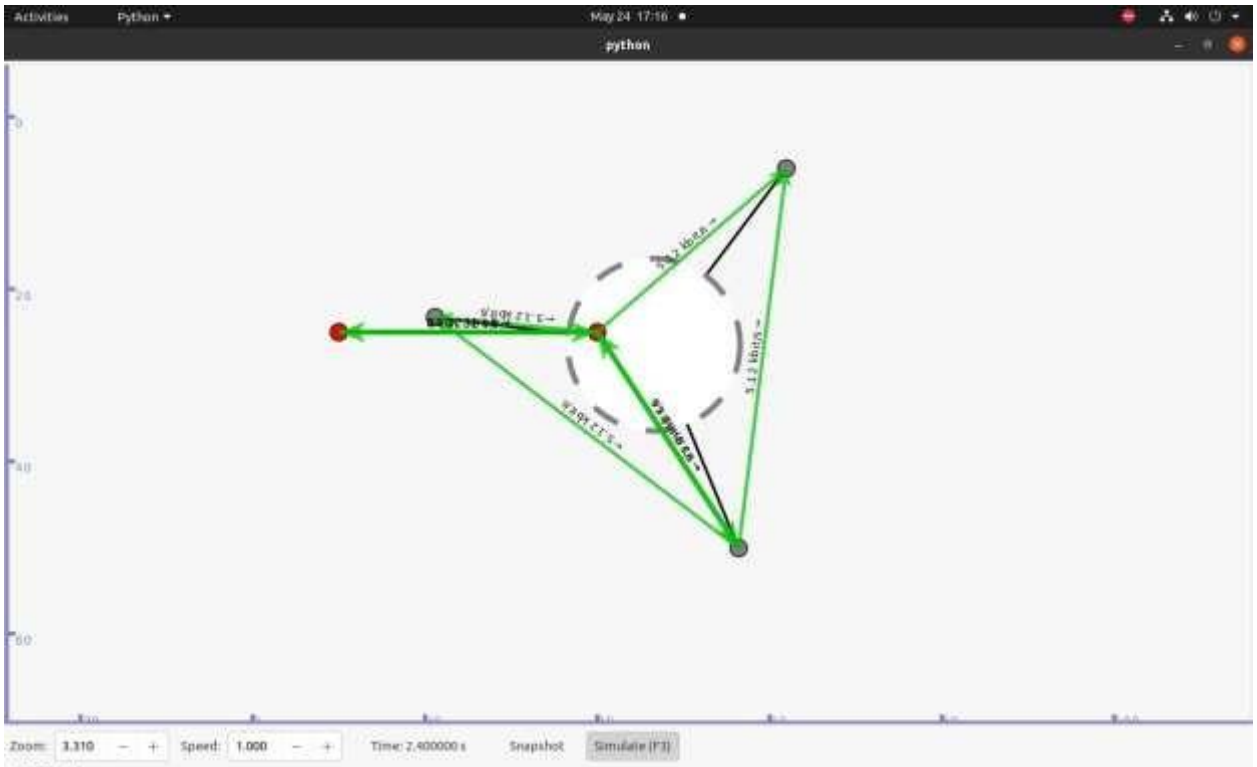
Output:

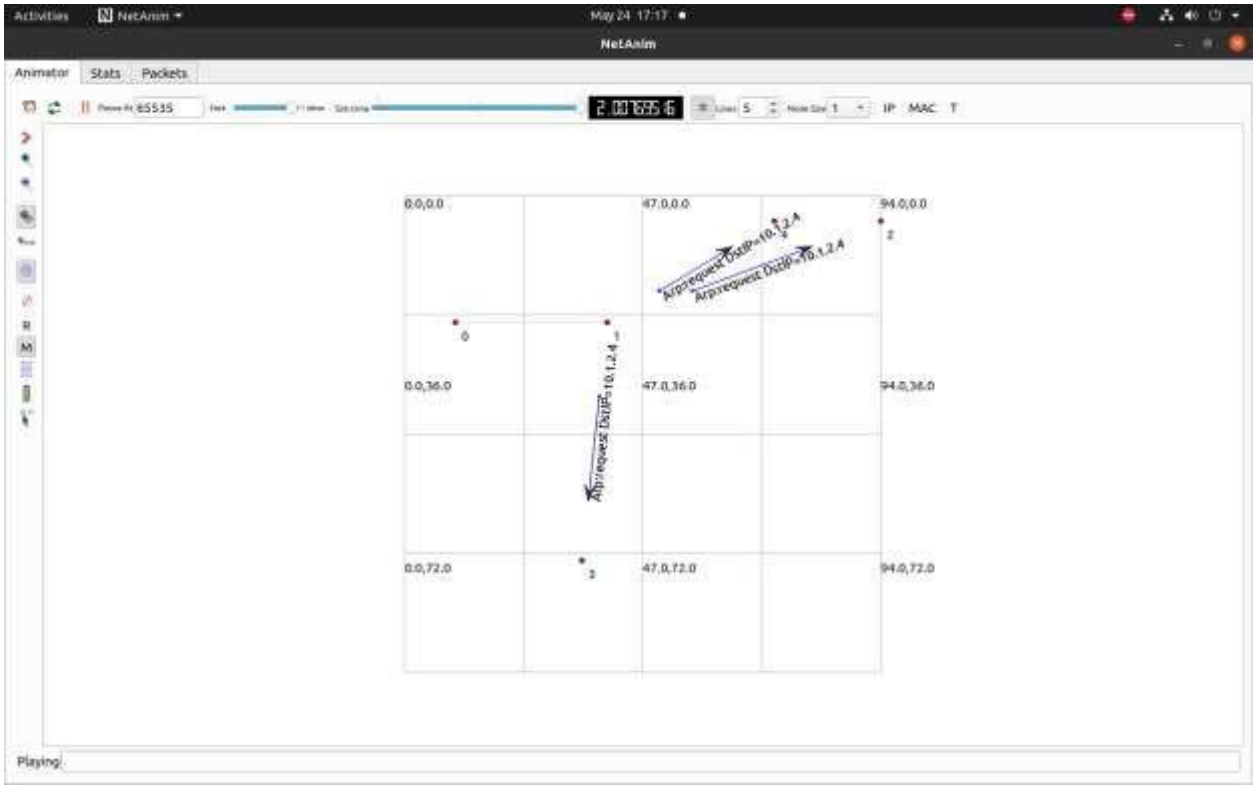


Roll No:-C22078
Name:-Pratik Vilas Mestry

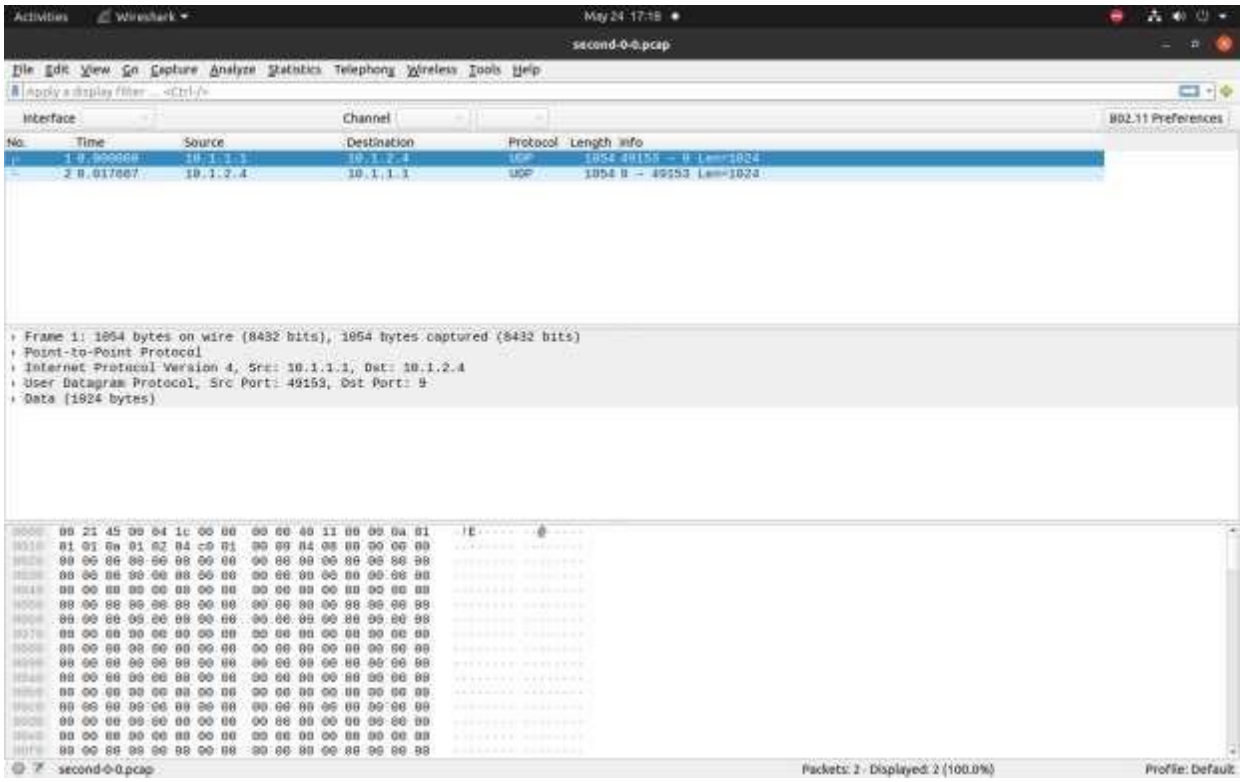


Roll No:-C22078
Name:-Pratik Vilas Mestry





Roll No:-C22078
Name:-Pratik Vilas Mestry



Roll No:-C22078
Name:-Pratik Vilas Mestry

Practical No. 7

Aim: Program to simulate mesh topology.

Code:

MESH:

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * Copyright (c) 2008,2009 IITP RAS
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * Author: Kirill Andreev <andreev@iitp.ru>
 *
 * By default this script creates m_xSize * m_ySize square grid topology with
 * IEEE802.11s stack installed at each node with peering management
 * and HWMP protocol.
 * The side of the square cell is defined by m_step parameter.
 * When topology is created, UDP ping is installed to opposite corners
 * by diagonals. packet size of the UDP ping and interval between two
 * successive packets is configurable.
 *
 *  m_xSize * step
 *  |<.....>|
 *   step
 *  |<--->|
 *  * --- * --- * <---Ping sink  _
 *  |\  |  /|           ^
 *  |  \|/  |           |
 *  * --- * --- * m_ySize * step |
 *  |  /|\  |           |
 *  |/  |  \|           |
 *  * --- * --- *           _
```

```
 *  ^ Ping source
 *
 *  See also MeshTest::Configure to read more about configurable
 *  parameters.
 */

#include <iostream>
#include <sstream>
#include <fstream>
#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mesh-module.h"
#include "ns3/mobility-module.h"
#include "ns3/mesh-helper.h"
#include "ns3/yans-wifi-helper.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("TestMeshScript");

/**
 * \ingroup mesh
 * \brief MeshTest class
 */
class MeshTest
{
public:
  /// Init test
  MeshTest ();
  /**
   * Configure test from command line arguments
   *
   * \param argc command line argument count
   * \param argv command line arguments
   */
  void Configure (int argc, char ** argv);
  /**
   * Run test
   * \returns the test status
   */
  int Run ();
```

```
private:
  int  m_xSize; ///< X size int
  m_ySize; ///< Y size double
  m_step; ///< step
  double    m_randomStart; ///< random start
  double    m_totalTime; ///< total time
  double    m_packetInterval; ///< packet interval
  uint16_t m_packetSize; ///< packet size
  uint32_t m_nIfaces; ///< number interfaces
  bool      m_chan; ///< channel
  bool      m_pcap; ///< PCAP
  bool      m_ascii; ///< ASCII
  std::string m_stack; ///< stack
  std::string m_root; ///< root
  /// List of network nodes
  NodeContainer nodes;
  /// List of all mesh point devices
  NetDeviceContainer meshDevices;
  /// Addresses of interfaces:
  Ipv4InterfaceContainer interfaces;
  /// MeshHelper. Report is not static methods
  MeshHelper mesh;



private:
  /// Create nodes and setup their mobility

  void CreateNodes ();
  /// Install internet m_stack on nodes
  void InstallInternetStack ();
  /// Install applications
  void InstallApplication ();
  /// Print mesh devices diagnostics
  void Report ();
};
MeshTest::MeshTest () :
  m_xSize (3),
  m_ySize (3),
  m_step (100.0),
  m_randomStart (0.1),
  m_totalTime (100.0),
  m_packetInterval (0.1),
  m_packetSize (1024),
  m_nIfaces (1),
```

```
  m_chan  (true),
  m_pcap (false),
  m_ascii (false),
  m_stack ("ns3::Dot11sStack"),
  m_root ("ff:ff:ff:ff:ff:ff")
{
}
void
MeshTest::Configure (int argc, char *argv[])
{
  CommandLine cmd (__FILE__);
  cmd.AddValue ("x-size", "Number of nodes in a row grid", m_xSize);
  cmd.AddValue ("y-size", "Number of rows in a grid", m_ySize);
  cmd.AddValue ("step",  "Size of edge in our grid (meters)", m_step);
  // Avoid starting all mesh nodes at the same time (beacons may collide)
  cmd.AddValue ("start",  "Maximum random start delay for beacon jitter (sec)", m_randomStart);
  cmd.AddValue ("time",  "Simulation time (sec)", m_totalTime);
  cmd.AddValue ("packet-interval",  "Interval between packets in UDP ping (sec)", m_packetInterval);
  cmd.AddValue ("packet-size",  "Size of packets in UDP ping (bytes)", m_packetSize);
  cmd.AddValue ("interfaces", "Number of radio interfaces used by each mesh point", m_nIfaces);
  cmd.AddValue ("channels", "Use different frequency channels for different interfaces", m_chan);
  cmd.AddValue ("pcap", "Enable PCAP traces on interfaces", m_pcap);
  cmd.AddValue ("ascii",  "Enable Ascii traces on interfaces", m_ascii);
  cmd.AddValue ("stack",  "Type of protocol stack. ns3::Dot11sStack by default", m_stack);



  cmd.AddValue ("root", "Mac address of root mesh point in HWMP", m_root);

  cmd.Parse (argc, argv);
  NS_LOG_DEBUG ("Grid:" << m_xSize << "*" << m_ySize);
  NS_LOG_DEBUG ("Simulation time: " << m_totalTime << " s");
  if (m_ascii)
    {
      PacketMetadata::Enable ();
    }
}
void
MeshTest::CreateNodes ()
{
  /*
   * Create m_ySize*m_xSize stations to form a grid topology
   */
  nodes.Create (m_ySize*m_xSize);
```

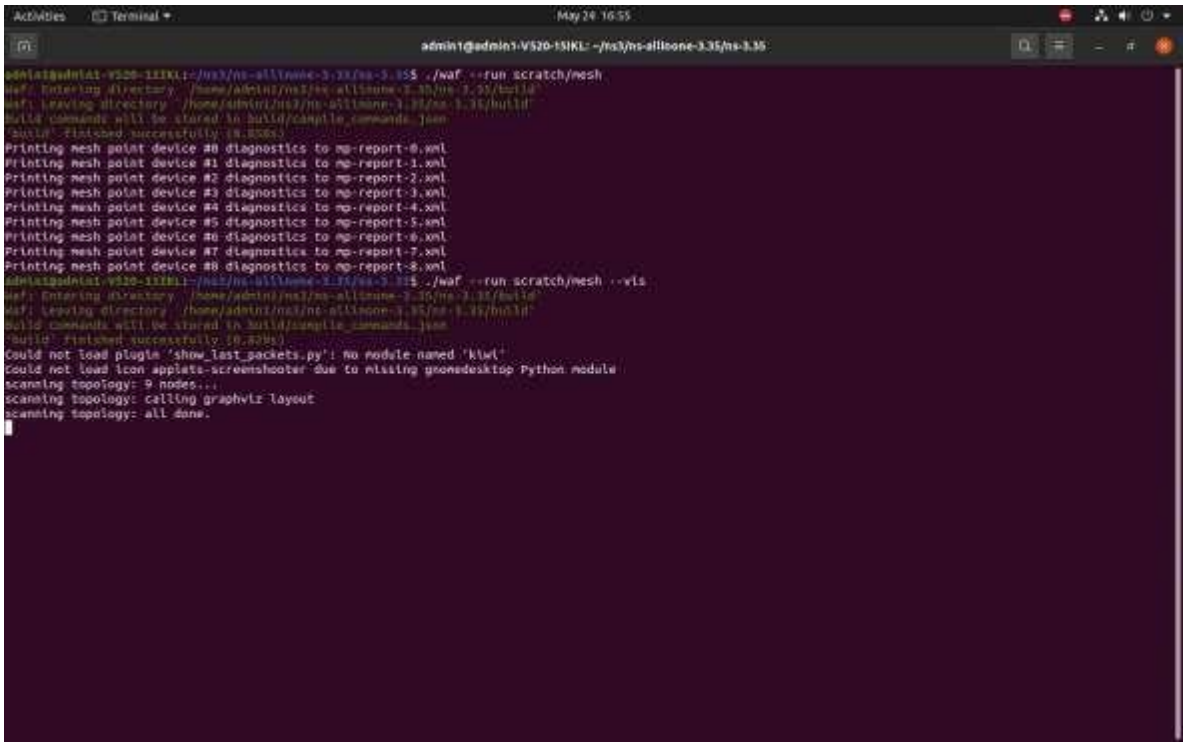
```
// Configure YansWifiChannel
YansWifiPhyHelper wifiPhy;
YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
wifiPhy.SetChannel (wifiChannel.Create ());
/*
 * Create mesh helper and set stack installer to it
 * Stack installer creates all needed protocols and install them to
 * mesh point device
 */
mesh = MeshHelper::Default ();
if (!Mac48Address (m_root.c_str ()).IsBroadcast ())
  {
    mesh.SetStackInstaller (m_stack, "Root", Mac48AddressValue (Mac48Address (m_root.c_str ())));
  }
else
  {
    //If root is not set, we do not use "Root" attribute, because it
    //is specified only for 11s
    mesh.SetStackInstaller (m_stack);
  }
if (m_chan)
  {
    mesh.SetSpreadInterfaceChannels (MeshHelper::SPREAD_CHANNELS);
  }
else
  {
    mesh.SetSpreadInterfaceChannels (MeshHelper::ZERO_CHANNEL);
  }
mesh.SetMacType ("RandomStart", TimeValue (Seconds (m_randomStart)));
// Set number of interfaces - default is single-interface mesh point

mesh.SetNumberOfInterfaces (m_nIfaces);
// Install protocols and return container if MeshPointDevices
meshDevices = mesh.Install (wifiPhy, nodes);
// Setup mobility - static grid topology
MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                               "MinX", DoubleValue (0.0),
                               "MinY", DoubleValue (0.0),
                               "DeltaX", DoubleValue (m_step),
                               "DeltaY", DoubleValue (m_step),
                               "GridWidth", UintegerValue (m_xSize),
                               "LayoutType", StringValue ("RowFirst"));
```

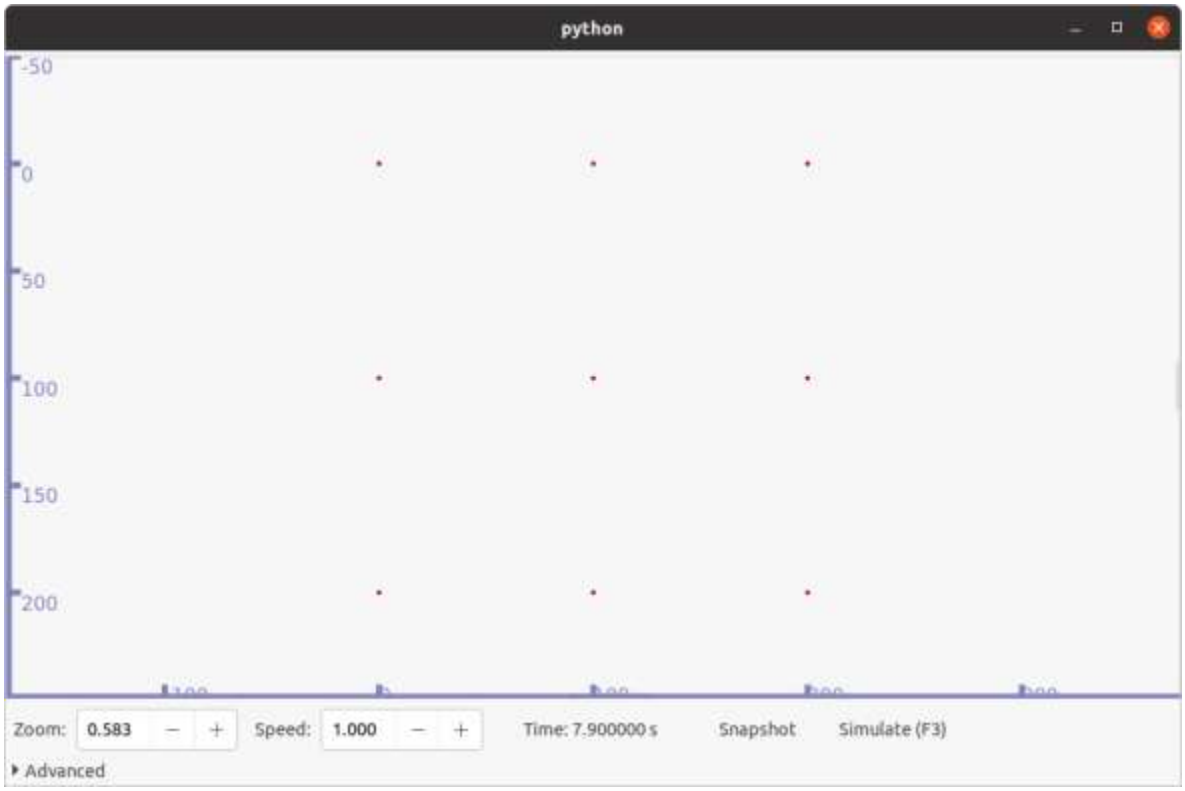
```
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility.Install (nodes);
  if (m_pcap)
    wifiPhy.EnablePcapAll (std::string ("mp-"));
  if (m_ascii)
    {
      AsciiTraceHelper ascii;
      wifiPhy.EnableAsciiAll (ascii.CreateFileStream ("mesh.tr"));
    }
}
void
MeshTest::InstallInternetStack ()
{
  InternetStackHelper internetStack;
  internetStack.Install (nodes);
  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");
  interfaces = address.Assign (meshDevices);
}
void
MeshTest::InstallApplication ()
{
  UdpEchoServerHelper echoServer (9);
  ApplicationContainer serverApps = echoServer.Install (nodes.Get (0));
  serverApps.Start (Seconds (0.0));
  serverApps.Stop (Seconds (m_totalTime));

  UdpEchoClientHelper echoClient (interfaces.GetAddress (0),
9); echoClient.SetAttribute ("MaxPackets", UintegerValue
((uint32_t)(m_totalTime*(1/m_packetInterval))));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (m_packetInterval)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (m_packetSize));
  ApplicationContainer clientApps = echoClient.Install (nodes.Get (m_xSize*m_ySize-1));
  clientApps.Start (Seconds (0.0));
  clientApps.Stop (Seconds (m_totalTime));
}
int
MeshTest::Run ()
{
  CreateNodes ();
  InstallInternetStack ();
  InstallApplication ();
  Simulator::Schedule (Seconds (m_totalTime), &MeshTest::Report, this);
```

```
  Simulator::Stop (Seconds (m_totalTime));
  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
void
MeshTest::Report ()
{
  unsigned n (0);
  for (NetDeviceContainer::Iterator i = meshDevices.Begin (); i != meshDevices.End (); ++i, ++n)
    {
      std::ostringstream os;
      os << "mp-report-" << n << ".xml";
      std::cerr << "Printing mesh point device #" << n << " diagnostics to " << os.str () << "\n";
      std::ofstream of;
      of.open (os.str ().c_str ());
      if (!of.is_open ())
        {
          std::cerr << "Error: Can't open file " << os.str () << "\n";
          return;
        }
      mesh.Report (*i, of);
      of.close ();
    }
}
Int
main (int argc, char *argv[])
{
  MeshTest t;
  t.Configure (argc, argv);
  return t.Run ();
```

Roll No:-C22078
Name:-Pratik Vilas Mestry

Output:





Roll No:-C22078
Name:-Pratik Vilas Mestry

Practical No. 8

Aim: Program to simulate hybrid topology.

Code:

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

#include <fstream>
#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"
//netanimation
#include "ns3/netanim-module.h"
#include "ns3/mobility-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FifthScriptExample");

//
// ============================================================================
//
//       node 0                 node 1
// +----------------+    +----------------+
// |    ns-3 TCP    |    |    ns-3 TCP    |
// +----------------+    +----------------+
// |    10.1.1.1    |    |    10.1.1.2    |
// +----------------+    +----------------+
```

```
//   | point-to-point |    | point-to-point |
//   +----------------+    +----------------+
//          |                 |
//          +-----------------+
//              5 Mbps, 2 ms
//
//
// We want to look at changes in the ns-3 TCP congestion window.  We need
// to crank up a flow and hook the CongestionWindow attribute on the socket
// of the sender.  Normally one would use an on-off application to generate a
// flow, but this has a couple of problems.  First, the socket of the on-off
// application is not created until Application Start time, so we wouldn't be
// able to hook the socket (now) at configuration time.  Second, even if we
// could arrange a call after start time, the socket is not public so we
// couldn't get at it.
//
// So, we can cook up a simple version of the on-off application that does what
// we want.  On the plus side we don't need all of the complexity of the on-off
// application.  On the minus side, we don't have a helper, so we have to get
// a little more involved in the details, but this is trivial.
//
// So first, we create a socket and do the trace connect on it; then we pass
// this socket into the constructor of our simple application which we then
// install in the source node.
//
============================================================================
//
class MyApp : public Application
{
public:

  MyApp ();
  virtual ~MyApp();

  void Setup (Ptr<Socket> socket, Address address, uint32_t packetSize, uint32_t nPackets, DataRate
dataRate);

private:
  virtual void StartApplication (void);
  virtual void StopApplication (void);

  void ScheduleTx (void);
```

```
  void SendPacket (void);

  Ptr<Socket>     m_socket;
  Address         m_peer;
  uint32_t        m_packetSize;
  uint32_t        m_nPackets;
  DataRate        m_dataRate;
  EventId         m_sendEvent;
  bool            m_running;
  uint32_t        m_packetsSent;
};

MyApp::MyApp ()
  : m_socket (0),
    m_peer (),
    m_packetSize (0),
    m_nPackets (0),
    m_dataRate (0),
    m_sendEvent (),
    m_running (false),
    m_packetsSent (0)
{
}

MyApp::~MyApp()
{
  m_socket = 0;
}

void
MyApp::Setup (Ptr<Socket> socket, Address address, uint32_t packetSize, uint32_t nPackets, DataRate
dataRate)
{
  m_socket = socket;
  m_peer = address;
  m_packetSize = packetSize;
  m_nPackets = nPackets;
  m_dataRate = dataRate;
}

void
MyApp::StartApplication (void)
{
```

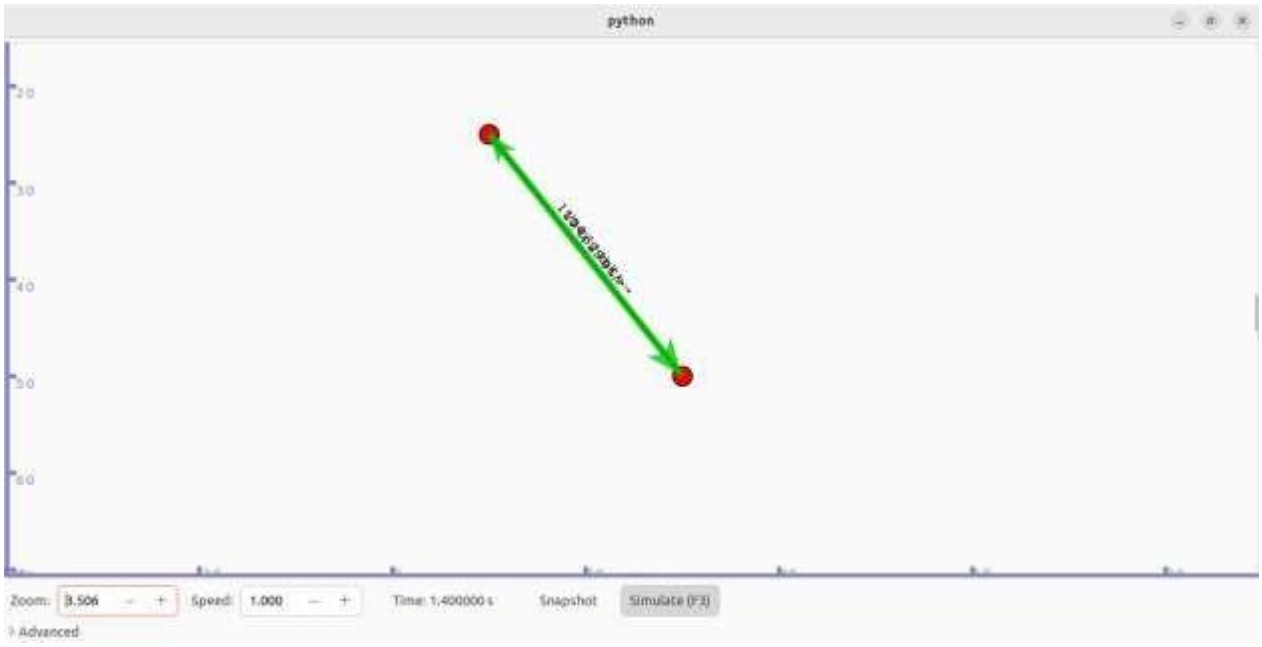
```
  m_running = true;
  m_packetsSent = 0;
  m_socket->Bind ();
  m_socket->Connect (m_peer);
  SendPacket ();
}

void
MyApp::StopApplication (void)
{
  m_running = false;

  if (m_sendEvent.IsRunning ())
    {
      Simulator::Cancel (m_sendEvent);
    }

  if (m_socket)
    {
      m_socket->Close ();
    }
}

void
MyApp::SendPacket (void)
{
  Ptr<Packet> packet = Create<Packet> (m_packetSize);
  m_socket->Send (packet);

  if (++m_packetsSent < m_nPackets)
    {
      ScheduleTx ();
    }
}

void
MyApp::ScheduleTx (void)
{
  if (m_running)
    {
      Time tNext (Seconds (m_packetSize * 8 / static_cast<double> (m_dataRate.GetBitRate ())));
      m_sendEvent = Simulator::Schedule (tNext, &MyApp::SendPacket, this);
    }}
```

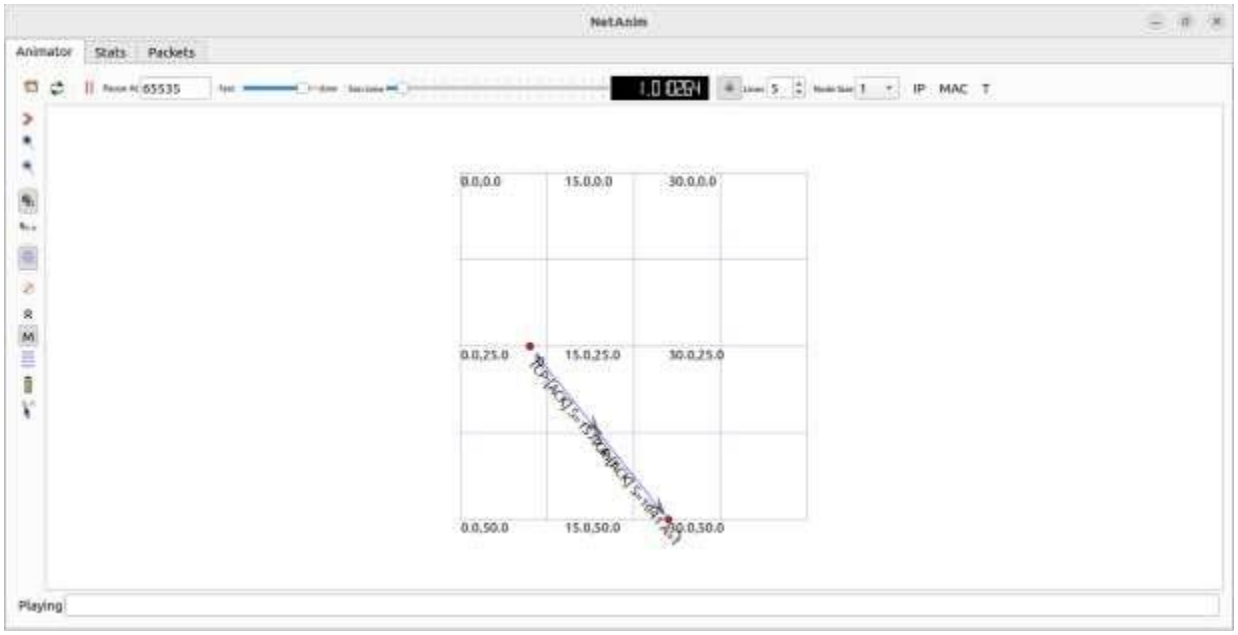
```
static void
CwndChange (uint32_t oldCwnd, uint32_t newCwnd)
{
  NS_LOG_UNCOND (Simulator::Now ().GetSeconds () << "\t" << newCwnd);
}

static void
RxDrop (Ptr<const Packet> p)
{
  NS_LOG_UNCOND ("RxDrop at " << Simulator::Now ().GetSeconds ());
}

int
main (int argc, char *argv[])
{
  CommandLine cmd (_FILE_);
  cmd.Parse (argc, argv);

  NodeContainer nodes;
  nodes.Create (2);

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);

  Ptr<RateErrorModel> em = CreateObject<RateErrorModel> ();
  em->SetAttribute ("ErrorRate", DoubleValue (0.00001));
  devices.Get (1)->SetAttribute ("ReceiveErrorModel", PointerValue (em));

  InternetStackHelper stack;
  stack.Install (nodes);

  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.252");
  Ipv4InterfaceContainer interfaces = address.Assign (devices);

  uint16_t sinkPort = 8080;
  Address sinkAddress (InetSocketAddress (interfaces.GetAddress (1), sinkPort));
PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory", InetSocketAddress
(Ipv4Address::GetAny (), sinkPort));
```

```
  ApplicationContainer sinkApps = packetSinkHelper.Install (nodes.Get (1));
  sinkApps.Start (Seconds (0.));
  sinkApps.Stop (Seconds (20.));

  Ptr<Socket> ns3TcpSocket = Socket::CreateSocket (nodes.Get (0), TcpSocketFactory::GetTypeId ());
  ns3TcpSocket->TraceConnectWithoutContext ("CongestionWindow", MakeCallback (&CwndChange));

  Ptr<MyApp> app = CreateObject<MyApp> ();
  app->Setup (ns3TcpSocket, sinkAddress, 1040, 1000, DataRate ("1Mbps"));
  nodes.Get (0)->AddApplication (app);
  app->SetStartTime (Seconds (1.));
  app->SetStopTime (Seconds (20.));

  devices.Get (1)->TraceConnectWithoutContext ("PhyRxDrop", MakeCallback (&RxDrop));


  MobilityHelper mobility;
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(nodes);

  AnimationInterface anim("fifth.xml");
  AnimationInterface::SetConstantPosition(nodes.Get(0),10,25);
  AnimationInterface::SetConstantPosition(nodes.Get(1),30,50);
  anim.EnablePacketMetadata(true);

  pointToPoint.EnablePcapAll("fifth");
  Simulator::Stop (Seconds (20));
  Simulator::Run ();
  Simulator::Destroy ();

  return 0;
}
```

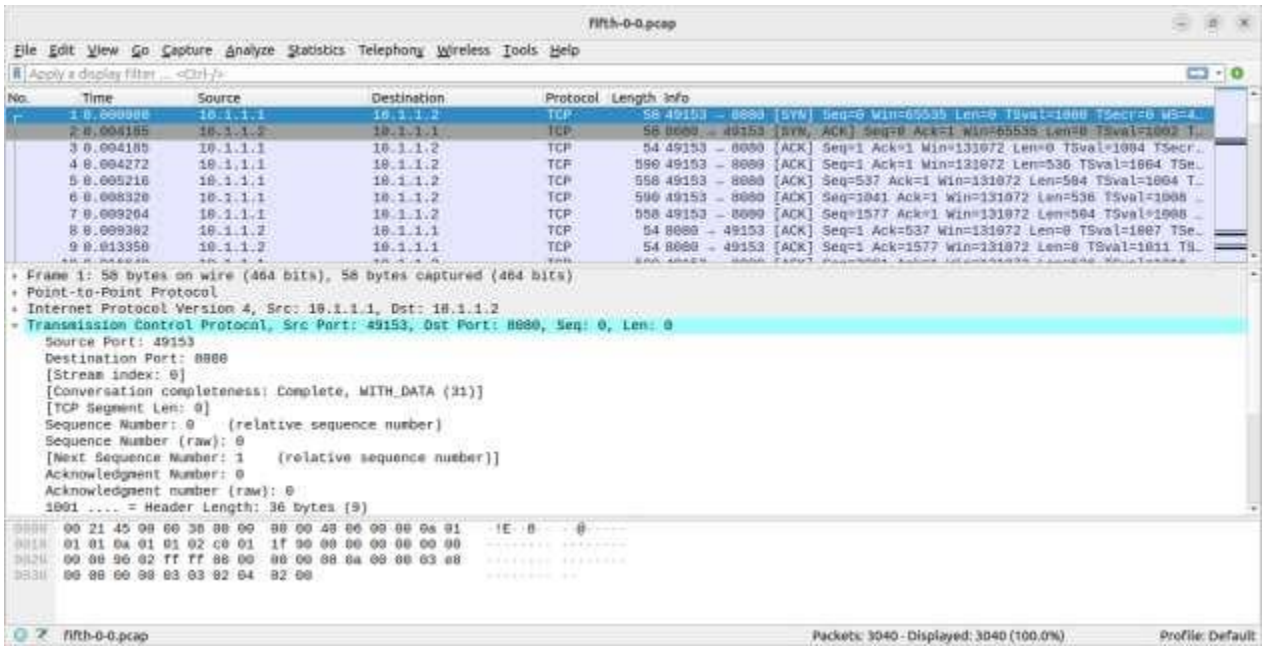
Roll No:-C22078
Name:-Pratik Vilas Mestry

Output:





Roll No:-C22078
Name:-Pratik Vilas Mestry



Roll No:-C22078
Name:-Pratik Vilas Mestry

Practical No. 9

Aim: Program to simulate UDP server.

Code:

udp_echo

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

// Network topology
//
//       n0    n1 n2   n3
//       |    |   |   |
//       ================
//             LAN
//
// - UDP flows from n0 to n1 and back
// - DropTail queues
// - Tracing of queues and packet receptions to file "udp-echo.tr"

#include <fstream>
#include "ns3/core-module.h"
#include "ns3/csma-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("UdpEchoExample");
```

Roll No:-C22078
Name:-Pratik Vilas Mestry

```
int
main (int argc, char *argv[])
{
//
// Users may find it convenient to turn on explicit debugging
// for selected modules; the below lines suggest how to do this
//
#if 0
  LogComponentEnable ("UdpEchoExample", LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_ALL);
LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_ALL);
#endif
//
// Allow the user to override any of the defaults and the above Bind() at
// run-time, via command-line arguments
//
  bool useV6 = false;
  Address serverAddress;

  CommandLine cmd (__FILE__);
  cmd.AddValue ("useIpv6", "Use Ipv6", useV6);
  cmd.Parse (argc, argv);

//
// Explicitly create the nodes required by the topology (shown above).
//
  NS_LOG_INFO ("Create nodes.");
  NodeContainer n;
  n.Create (4);

  InternetStackHelper internet;
  internet.Install (n);

  NS_LOG_INFO ("Create channels.");
//
// Explicitly create the channels required by the topology (shown above).
//
  CsmaHelper csma;
  csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (5000000)));
  csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
  csma.SetDeviceAttribute ("Mtu", UintegerValue (1400));
  NetDeviceContainer d = csma.Install (n);
```

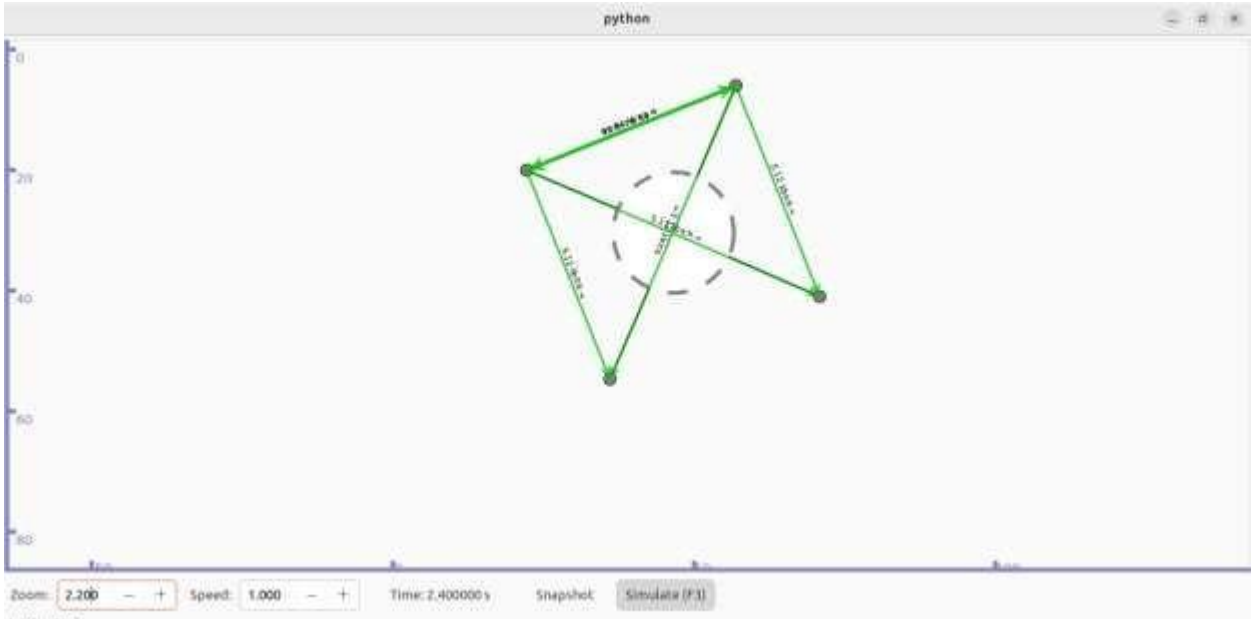
Roll No:-C22078
Name:-Pratik Vilas Mestry

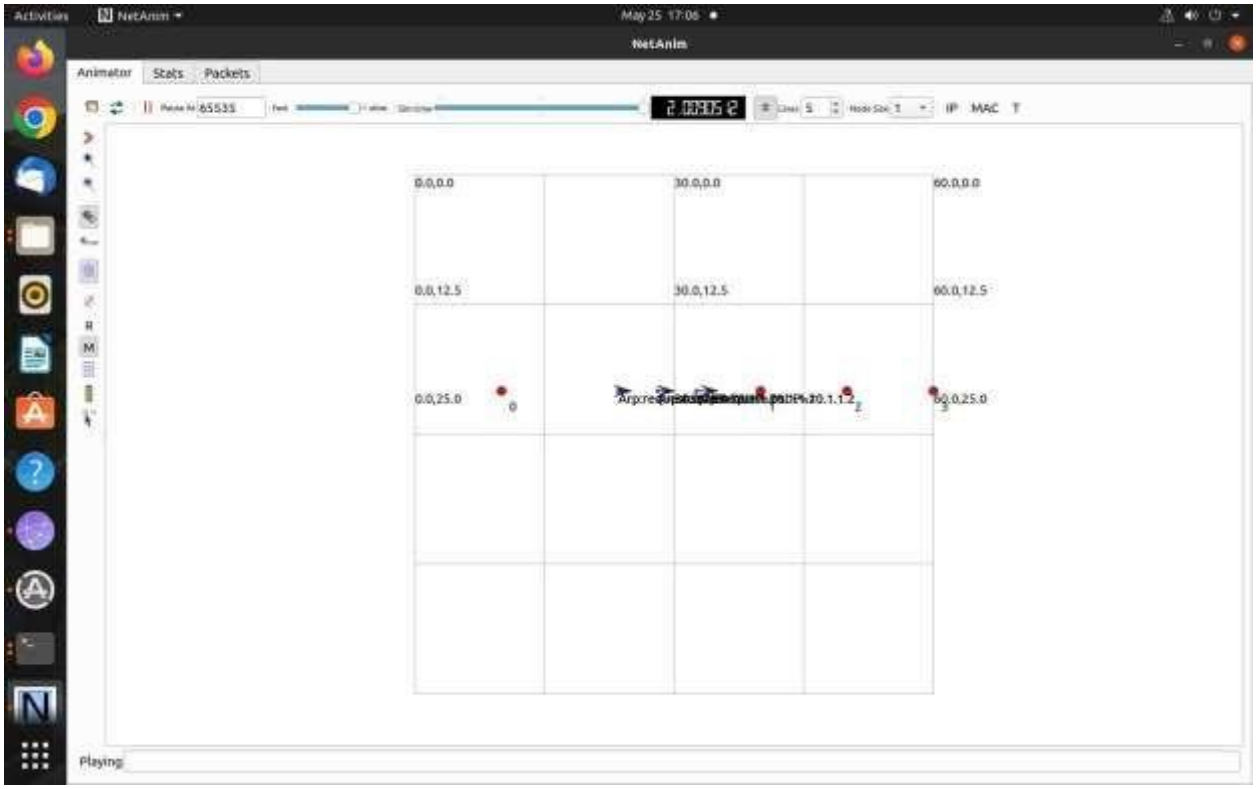
```
//
// We've got the "hardware" in place.  Now we need to add IP addresses.
//
  NS_LOG_INFO ("Assign IP Addresses.");
  if (useV6 == false)
    {
      Ipv4AddressHelper ipv4;
      ipv4.SetBase ("10.1.1.0", "255.255.255.0");
      Ipv4InterfaceContainer i = ipv4.Assign (d);
      serverAddress = Address(i.GetAddress (1));
    }
  else
    {
      Ipv6AddressHelper ipv6;
      ipv6.SetBase ("2001:0000:f00d:cafe::", Ipv6Prefix (64));
      Ipv6InterfaceContainer i6 = ipv6.Assign (d);
      serverAddress = Address(i6.GetAddress (1,1));
    }

  NS_LOG_INFO ("Create Applications.");
//
// Create a UdpEchoServer application on node one.
//

  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));

//
// Create a UdpEchoClient application to send UDP datagrams from node zero to
// node one.
//
  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
//
// Users may find it convenient to initialize echo packets with actual data;
// the below lines suggest how to do this
//
  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

  AsciiTraceHelper ascii;
  csma.EnableAsciiAll (ascii.CreateFileStream ("udp-echo.tr"));
  csma.EnablePcapAll ("udp-echo", false);

//
// Now, do the actual simulation.
//
  NS_LOG_INFO ("Run Simulation.");
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");
}
```

Roll No:-C22078
Name:-Pratik Vilas Mestry

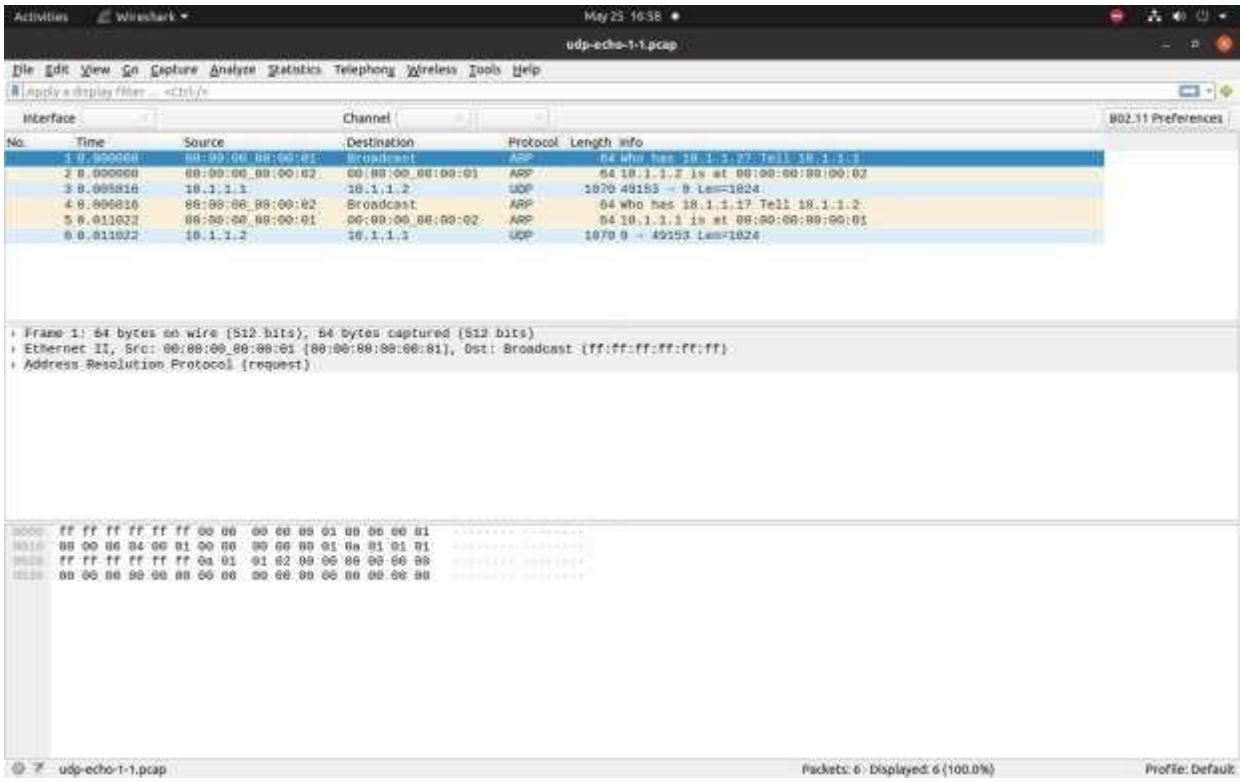
Output:





Roll No:-C22078
Name:-Pratik Vilas Mestry

Roll No:-C22078
Name:-Pratik Vilas Mestry



Roll No:-C22078
Name:-Pratik Vilas Mestry

Practical No. 10

Aim: Program to simulate UDP server client.

Code:

UDP-client-server

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * Copyright (c) 2009 INRIA
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * Author: Mohamed Amine Ismail <amine.ismail@sophia.inria.fr>
 */

// Network topology
//
//       n0    n1
//       |     |
//       =======
//         LAN (CSMA)
//
// - UDP flow from n0 to n1 of 1024 byte packets at intervals of 50 ms
//   - maximum of 320 packets sent (or limited by simulation duration)
//  - option to use IPv4 or IPv6 addressing
// - option to disable logging statements

#include <fstream>
#include "ns3/core-module.h"
#include "ns3/csma-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"
```

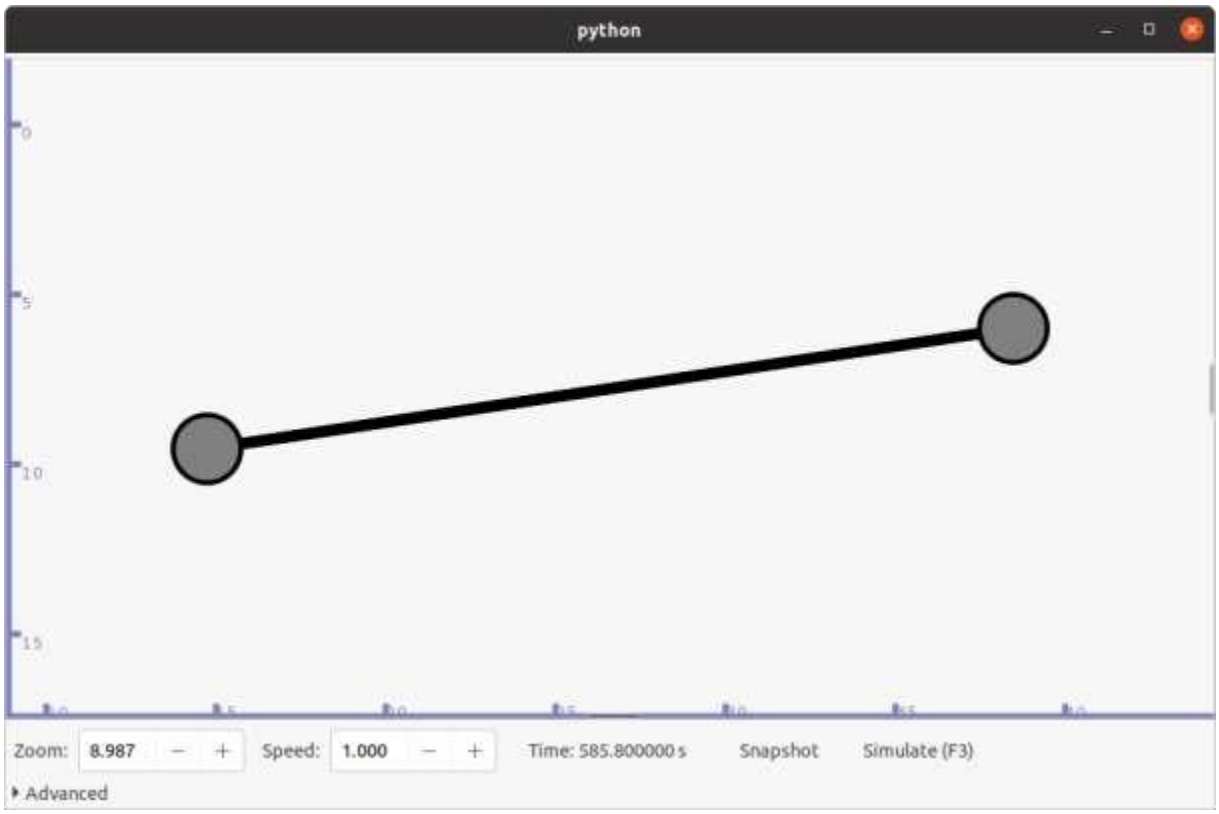
Roll No:-C22078
Name:-Pratik Vilas Mestry

```
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("UdpClientServerExample");

int
main (int argc, char *argv[])
{
 // Declare variables used in command-line arguments
 bool useV6 = false;
 bool logging = true;
 Address serverAddress;

 CommandLine cmd (__FILE__);
 cmd.AddValue ("useIpv6", "Use Ipv6", useV6);
 cmd.AddValue ("logging", "Enable logging", logging);
 cmd.Parse (argc, argv);

 if (logging)
  {
   LogComponentEnable ("UdpClient", LOG_LEVEL_INFO);
   LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);

  }

 NS_LOG_INFO ("Create nodes in above topology.");
 NodeContainer n;
 n.Create (2);

 InternetStackHelper internet;
 internet.Install (n);

 NS_LOG_INFO ("Create channel between the two nodes.");
 CsmaHelper csma;
 csma.SetChannelAttribute ("DataRate", DataRateValue (DataRate (5000000)));
 csma.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (2)));
 csma.SetDeviceAttribute ("Mtu", UintegerValue (1400));
 NetDeviceContainer d = csma.Install (n);

 NS_LOG_INFO ("Assign IP Addresses.");
 if (useV6 == false)
  {
   Ipv4AddressHelper ipv4;
   ipv4.SetBase ("10.1.1.0", "255.255.255.0");
   Ipv4InterfaceContainer i = ipv4.Assign (d);
```

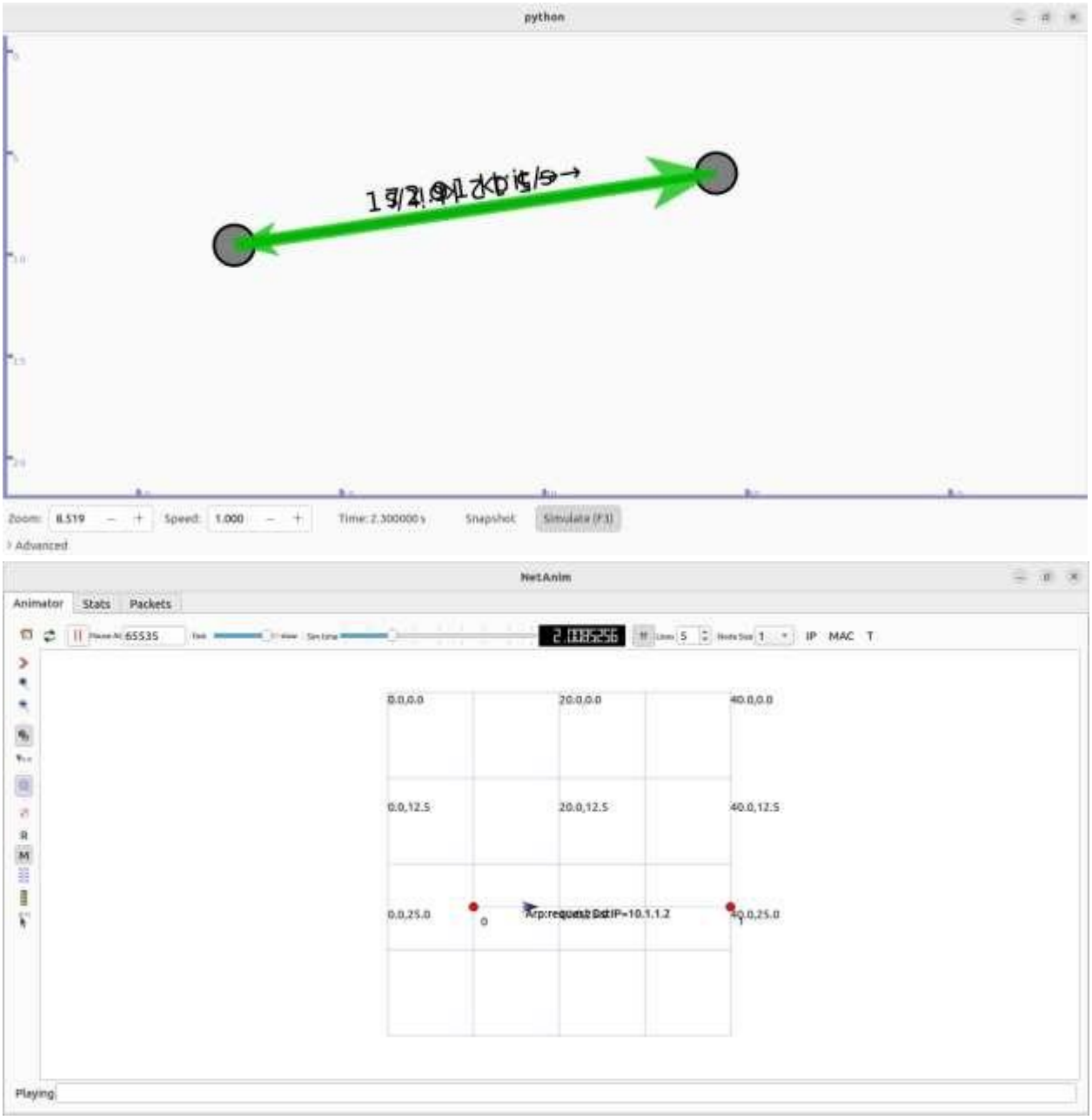
```
    serverAddress = Address (i.GetAddress (1));

  }
 else
  {
   Ipv6AddressHelper ipv6;
   ipv6.SetBase ("2001:0000:f00d:cafe::", Ipv6Prefix (64));
   Ipv6InterfaceContainer i6 = ipv6.Assign (d);
   serverAddress = Address(i6.GetAddress (1,1));
  }

 NS_LOG_INFO ("Create UdpServer application on node 1.");
 uint16_t port = 4000;
 UdpServerHelper server (port);
 ApplicationContainer apps = server.Install (n.Get (1));
 apps.Start (Seconds (1.0));
 apps.Stop (Seconds (10.0));

 NS_LOG_INFO ("Create UdpClient application on node 0 to send to node 1.");
 uint32_t MaxPacketSize = 1024;
 Time interPacketInterval = Seconds (0.05);
 uint32_t maxPacketCount = 320;
 UdpClientHelper client (serverAddress, port);

 client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount));
 client.SetAttribute ("Interval", TimeValue (interPacketInterval));
 client.SetAttribute ("PacketSize", UintegerValue (MaxPacketSize));
 apps = client.Install (n.Get (0));
 apps.Start (Seconds (2.0));
 apps.Stop (Seconds (10.0));

 NS_LOG_INFO ("Run Simulation.");
 Simulator::Run  ();
 Simulator::Destroy ();
 NS_LOG_INFO ("Done.");
}
```

Roll No:-C22078
Name:-Pratik Vilas Mestry

Output:



Roll No:-C22078
Name:-Pratik Vilas Mestry



Roll No:-C22078
Name:-Pratik Vilas Mestry

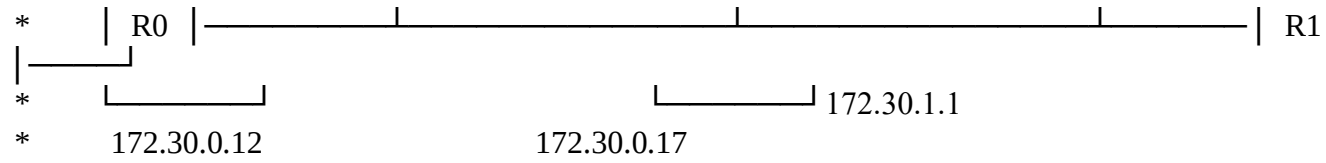
Practical No. 11

Aim: Program to simulate DHCP server and n clients.

Code:

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * Copyright (c) 2011 UPB
 * Copyright (c) 2017 NITK Surathkal
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * Author: Radu Lupu <rlupu@elcom.pub.ro>
 *         Ankit Deepak <adadeepak8@gmail.com>
 *         Deepti Rajagopal <deeptir96@gmail.com>
 *
 */


/*
 * Network layout:
 *
 * R0 is a DHCP server. The DHCP server announced R1 as the default router.
 * Nodes N1 will send UDP Echo packets to node A.
 *
 *
 *
 *            ┌------------------------------------------------┐
 *            | DHCP Clients                                   |
 *            |                          172.30.0.14           |
 *            |                          DHCP static           |
 *            |  ┌─────────┐   ┌─────────┐   ┌─────────┐       |
 *            |  │   N0    │   │   N1    │   │   N2    │   |   ┌─────────┐
 *            |  └─────────┘   └─────────┘   └─────────┘       |   │    A    │
 *            |    │           │           │       |     └─────────┘
 *            └----|-----------|-----------|-------┘       | 172.30.1.2
 *  DHCP Server    |           |           |               |
 *       ┌─────────┐       |           |           |     ┌─────────┐     |
```

Roll No:-C22078
Name:-Pratik Vilas Mestry




```
*
* Things to notice:
* 1) The routes in A are manually set to have R1 as the default router,
*    just because using a dynamic outing in this example is an overkill.
* 2) R1's address is set statically though the DHCP server helper interface.
*    This is useful to prevent address conflicts with the dynamic pool.
*    Not necessary if the DHCP pool is not conflicting with static addresses.
* 3) N2 has a dynamically-assigned, static address (i.e., a fixed address assigned via DHCP).
*
*/

#include "ns3/core-module.h"
#include "ns3/internet-apps-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"

//netanim code
#include "ns3/netanim-module.h"
#include "ns3/mobility-module.h"


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("DhcpExample");

int
main (int argc, char *argv[])
{
  CommandLine cmd (__FILE__);

  bool verbose = false;
  bool tracing = false;
  cmd.AddValue ("verbose", "turn on the logs", verbose);
  cmd.AddValue ("tracing", "turn on the tracing", tracing);

  cmd.Parse (argc, argv);

  // GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true));

  if (verbose)
    {
```

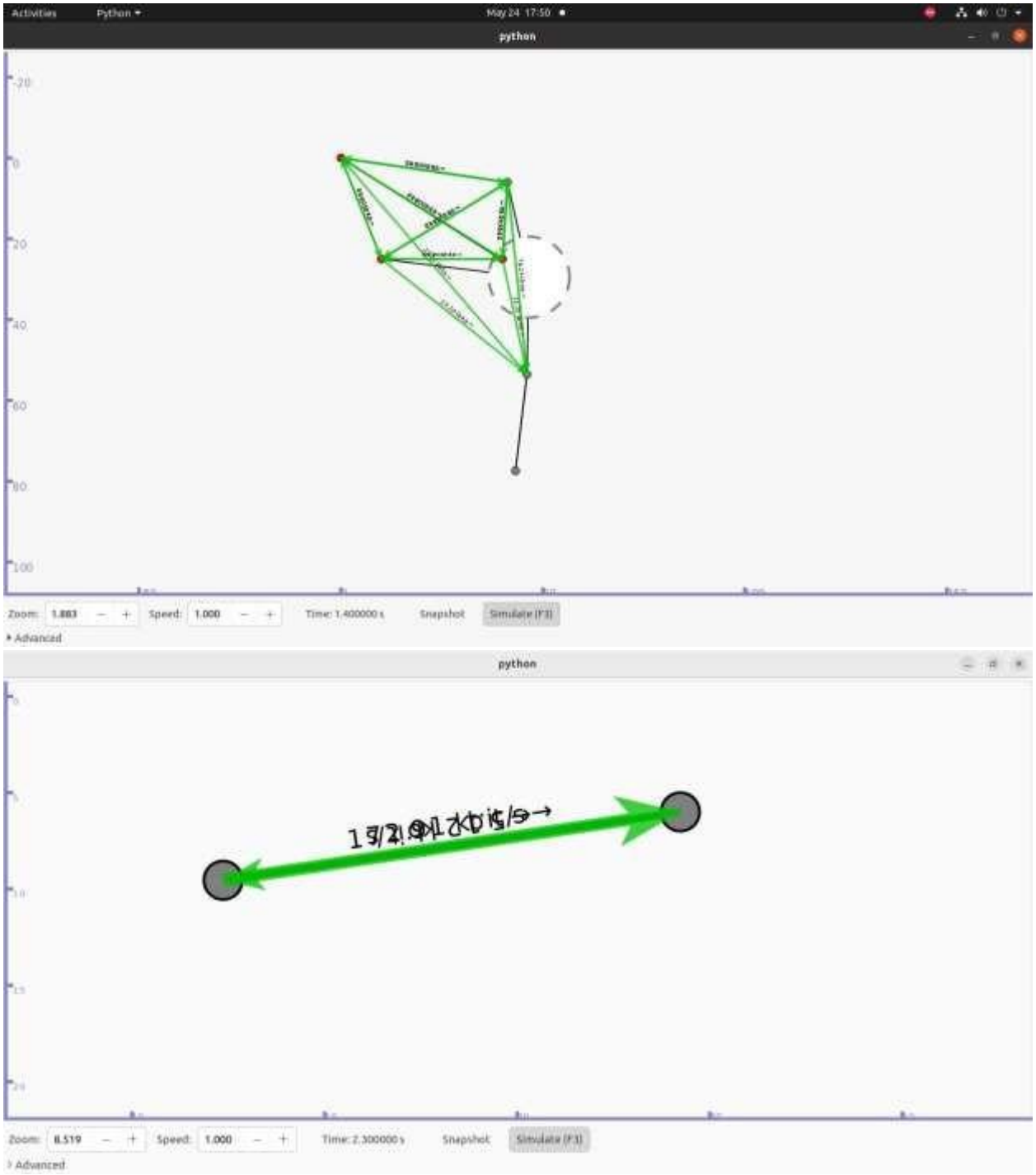
```
    LogComponentEnable ("DhcpServer", LOG_LEVEL_ALL);
    LogComponentEnable ("DhcpClient", LOG_LEVEL_ALL);
    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  }

Time stopTime = Seconds (20);

NS_LOG_INFO ("Create nodes.");
NodeContainer nodes;
NodeContainer router;
nodes.Create (3);
router.Create (2);

NodeContainer net (nodes, router);

NS_LOG_INFO ("Create channels.");
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("5Mbps"));
csma.SetChannelAttribute ("Delay", StringValue ("2ms"));
csma.SetDeviceAttribute ("Mtu", UintegerValue (1500));
NetDeviceContainer devNet = csma.Install (net);

NodeContainer p2pNodes;
p2pNodes.Add (net.Get (4));
p2pNodes.Create (1);

PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

NetDeviceContainer p2pDevices;
p2pDevices = pointToPoint.Install (p2pNodes);

InternetStackHelper tcpip;
tcpip.Install  (nodes);
tcpip.Install (router);
tcpip.Install (p2pNodes.Get (1));

Ipv4AddressHelper address;
address.SetBase ("172.30.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);

// manually add a routing entry because we don't want to add a dynamic routing
Ipv4StaticRoutingHelper ipv4RoutingHelper;
Ptr<Ipv4> ipv4Ptr = p2pNodes.Get (1)->GetObject<Ipv4> ();
```

```
  Ptr<Ipv4StaticRouting> staticRoutingA = ipv4RoutingHelper.GetStaticRouting (ipv4Ptr);
  staticRoutingA->AddNetworkRouteTo (Ipv4Address ("172.30.0.0"), Ipv4Mask ("/24"),
                                     Ipv4Address ("172.30.1.1"), 1);

  NS_LOG_INFO ("Setup the IP addresses and create DHCP applications.");
  DhcpHelper dhcpHelper;

  // The router must have a fixed IP.
  Ipv4InterfaceContainer fixedNodes = dhcpHelper.InstallFixedAddress (devNet.Get (4), Ipv4Address
("172.30.0.17"), Ipv4Mask ("/24"));
  // Not really necessary, IP forwarding is enabled by default in IPv4.
  fixedNodes.Get (0).first->SetAttribute ("IpForward", BooleanValue (true));

  // DHCP server
  ApplicationContainer dhcpServerApp = dhcpHelper.InstallDhcpServer (devNet.Get (3), Ipv4Address
("172.30.0.12"),
                                                                     Ipv4Address ("172.30.0.0"), Ipv4Mask ("/24"),
                                                                     Ipv4Address ("172.30.0.10"), Ipv4Address ("172.30.0.15"),
                                                                     Ipv4Address ("172.30.0.17"));

  // This is just to show how it can be done.
  DynamicCast<DhcpServer> (dhcpServerApp.Get (0))->AddStaticDhcpEntry (devNet.Get (2)-
>GetAddress (), Ipv4Address ("172.30.0.14"));

  dhcpServerApp.Start (Seconds (0.0));
  dhcpServerApp.Stop (stopTime);

  // DHCP clients
  NetDeviceContainer dhcpClientNetDevs;
  dhcpClientNetDevs.Add (devNet.Get (0));
  dhcpClientNetDevs.Add (devNet.Get (1));
  dhcpClientNetDevs.Add (devNet.Get (2));

  ApplicationContainer dhcpClients = dhcpHelper.InstallDhcpClient (dhcpClientNetDevs);
  dhcpClients.Start (Seconds (1.0));
  dhcpClients.Stop (stopTime);

  UdpEchoServerHelper echoServer (9);

  ApplicationContainer serverApps = echoServer.Install (p2pNodes.Get (1));
  serverApps.Start (Seconds (0.0));
  serverApps.Stop (stopTime);

  UdpEchoClientHelper echoClient (p2pInterfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (100));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
```

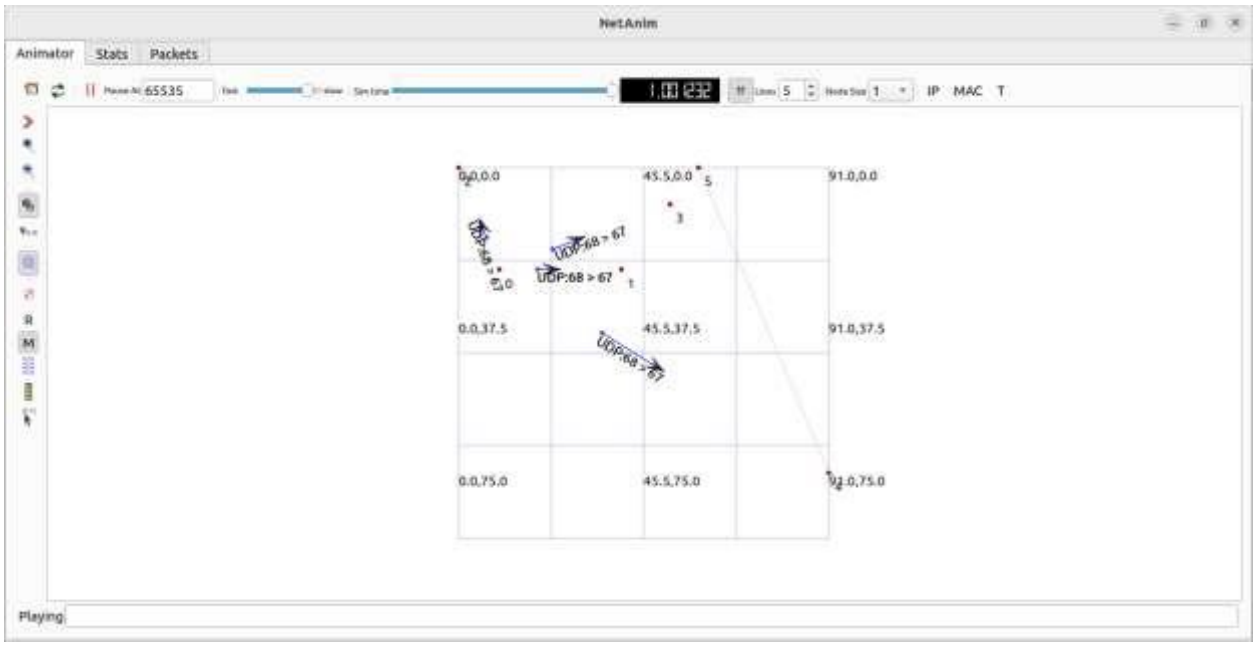
```
  ApplicationContainer clientApps = echoClient.Install (nodes.Get (1));
  clientApps.Start (Seconds (10.0));
  clientApps.Stop (stopTime);


  MobilityHelper mobility;
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(nodes);

  AnimationInterface anim("pranay.xml");
  AnimationInterface::SetConstantPosition(nodes.Get(0),10,25);
  AnimationInterface::SetConstantPosition(nodes.Get(1),40,25);
  anim.EnablePacketMetadata(true);



  Simulator::Stop (stopTime + Seconds (10.0));

  if (tracing)
    {
      csma.EnablePcapAll ("dhcp-csma");
      pointToPoint.EnablePcapAll ("dhcp-p2p");
    }

  NS_LOG_INFO ("Run Simulation.");
  Simulator::Run  ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");
}
```

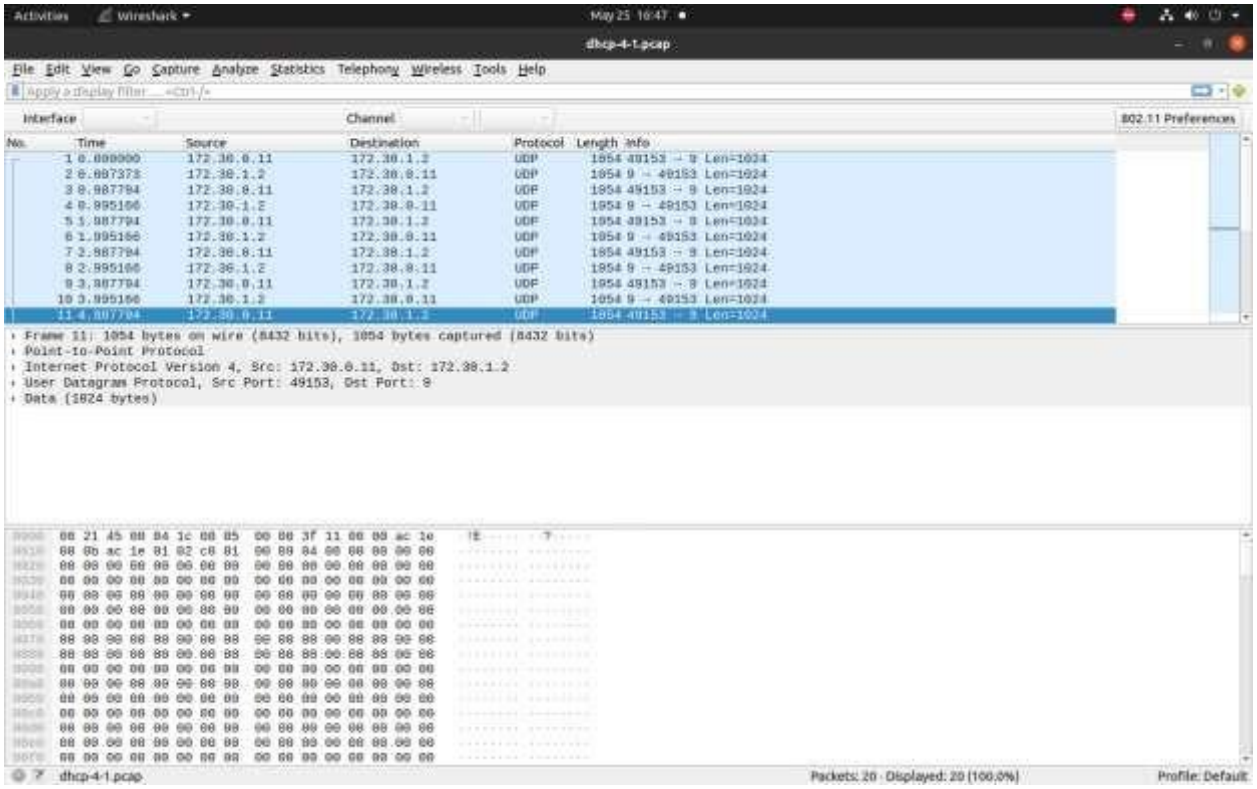
Roll No:-C22078
Name:-Pratik Vilas Mestry

Output:



Roll No:-C22078
Name:-Pratik Vilas Mestry





Roll No:-C22078
Name:-Pratik Vilas Mestry

Practical No. 12

Aim: Animate a simple network using NetAnim in Network Simulator.

Code:

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307  USA
 */

#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"

//netanimator
#include "ns3/netanim-module.h"
#include "ns3/mobility-module.h"

// Default Network Topology
//
//       10.1.1.0
// n0----------------n1
//    point-to-point
//
```

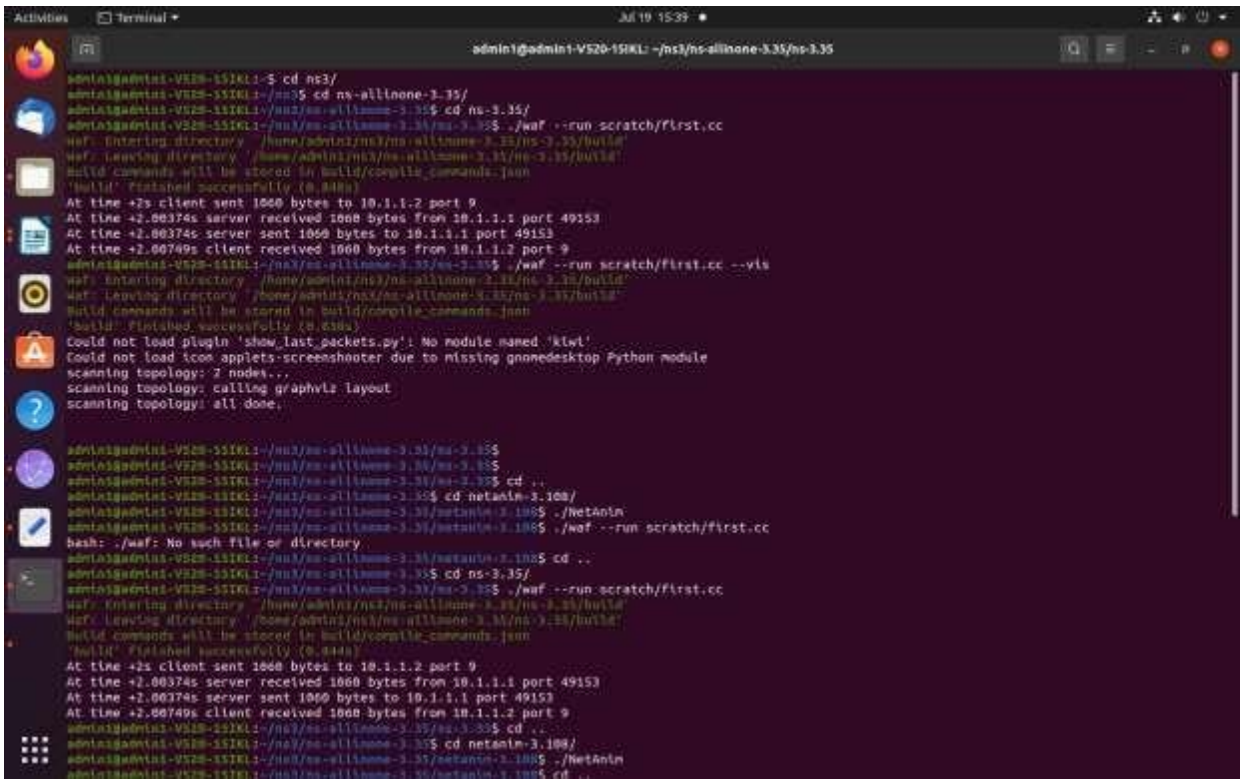
Roll No:-C22078
Name:-Pratik Vilas Mestry

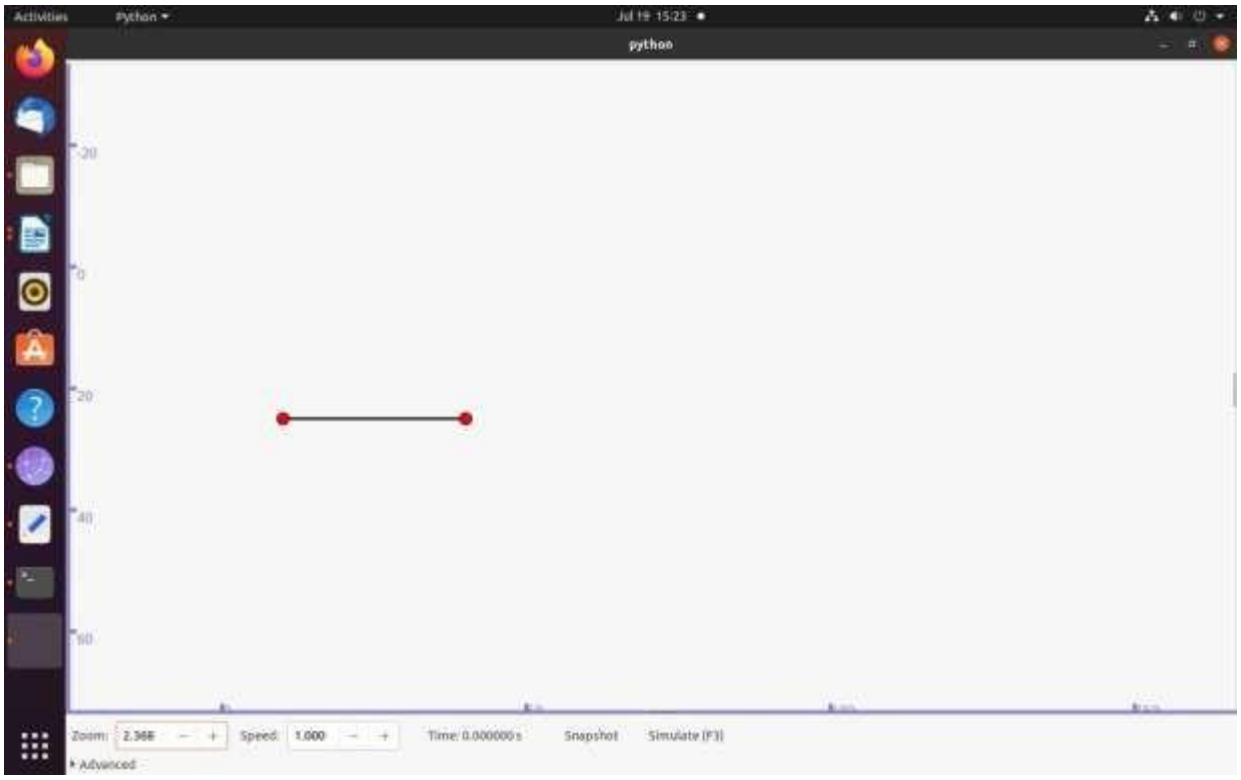
```
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

int
main (int argc, char *argv[])
{
  CommandLine cmd (_FILE_);
  cmd.Parse (argc, argv);

  Time::SetResolution (Time::NS);
  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

  NodeContainer nodes;
  nodes.Create (2);

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);

  InternetStackHelper stack;
  stack.Install (nodes);

  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");

  Ipv4InterfaceContainer interfaces = address.Assign (devices);

  UdpEchoServerHelper echoServer (9);

  ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));
```

```
  UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1060));
  ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  MobilityHelper mobility;
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(nodes);
  AnimationInterface anim("first.xml");
  AnimationInterface::SetConstantPosition(nodes.Get(0),10,25);
  AnimationInterface::SetConstantPosition(nodes.Get(1),40,25);

  anim.EnablePacketMetadata(true);

  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

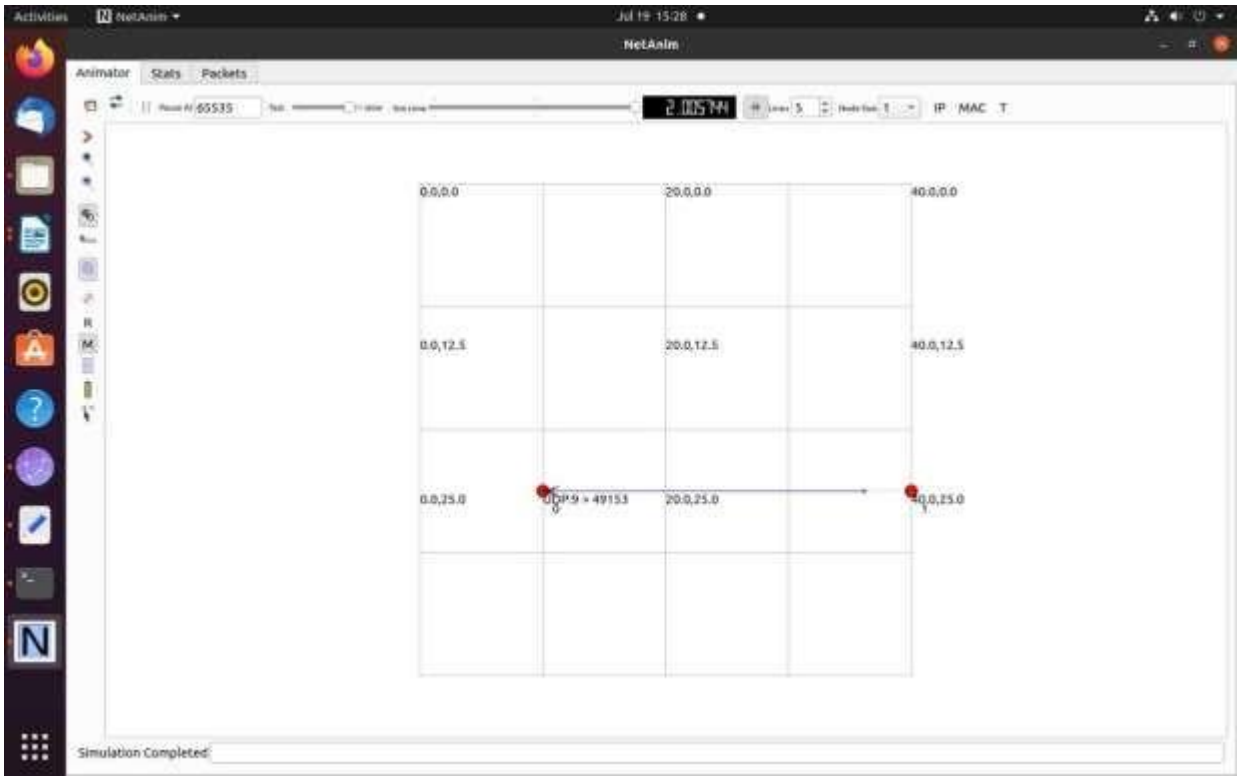
Roll No:-C22078
Name:-Pratik Vilas Mestry

Output:





Roll No:-C22078
Name:-Pratik Vilas Mestry



Roll No:-C22078
Name:-Pratik Vilas Mestry

Practical No. 13

Aim: Analyze the network traffic using Wire Shark.

Code:

```
#include "ns3/core-module.h" #include
"ns3/network-module.h"#include
"ns3/netanim-module.h"#include
"ns3/internet-module.h"

#include "ns3/point-to-point-module.h" #include
"ns3/applications-module.h" #include "ns3/point-to-point-
layout-module.h"#include "ns3/netanim-module.h"

#include "ns3/mobility-module.h"



// Network topology (default)
//

  //       n2 n3 n4              .
  //        \ | /                .
  //         \|/                 .
  //    n1--- n0---n5            .
  //         /|\                 .
  //        / | \                .
  //       n8 n7 n6              .
```

Roll No:-C22078
Name:-Pratik Vilas Mestry

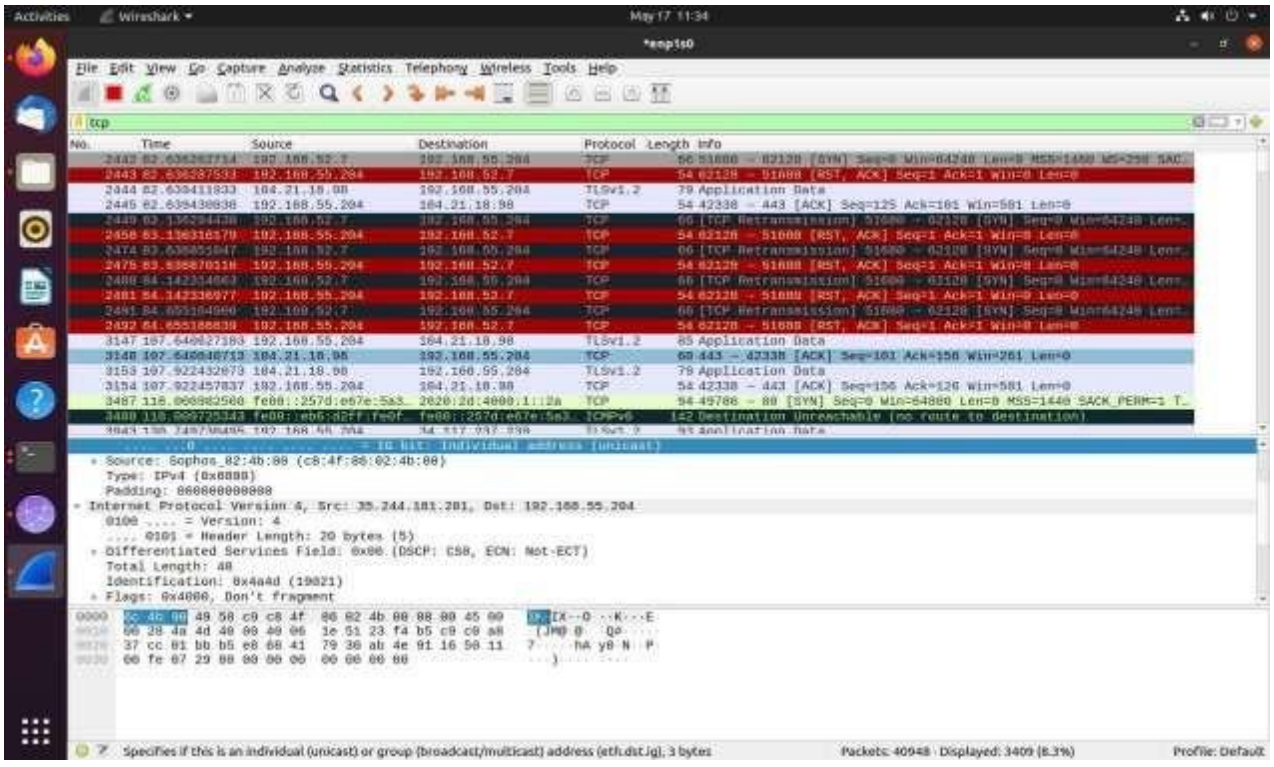
```
//
using namespace ns3; NS_LOG_COMPONENT_DEFINE
("Star");

int main (int argc, char *argv[])
{
        NodeContainer nodes;
        nodes.Create(9);

  //
  // Set up some default values for the simulation.
  //
 Config::SetDefault ("ns3::OnOffApplication::PacketSize", UintegerValue(137));
  // ??? try and stick 15kb/s into the data rate
 Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue("14kb/s"));
  //
 // Default number of nodes in the star. Overridable by command lineargument.
  //
  uint32_t nSpokes = 8; CommandLine
  cmd (__________________________FILE____);

 cmd.AddValue ("nSpokes", "Number of nodes to place in the star",nSpokes);
  cmd.Parse (argc, argv); NS_LOG_INFO ("Build star
  topology.");PointToPointHelper pointToPoint;

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

```
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
PointToPointStarHelper star (nSpokes, pointToPoint); NS_LOG_INFO ("Install
internet stack on all nodes."); InternetStackHelper internet;
star.InstallStack (internet); NS_LOG_INFO ("Assign
IP Addresses.");
star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0",
"255.255.255.0"));
NS_LOG_INFO ("Create applications.");
//
// Create a packet sink on the star "hub" to receive packets.
//
uint16_t port = 50000;
Address hubLocalAddress (InetSocketAddress (Ipv4Address::GetAny (),port));
PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory",hubLocalAddress);
ApplicationContainer hubApp = packetSinkHelper.Install (star.GetHub ());hubApp.Start
(Seconds (1.0));
hubApp.Stop (Seconds (10.0));
//
// Create OnOff applications to send TCP to the hub, one on each spokenode.
//
OnOffHelper onOffHelper ("ns3::TcpSocketFactory", Address ());
onOffHelper.SetAttribute ("OnTime", StringValue("ns3::ConstantRandomVariable[Constant=1]"));
```

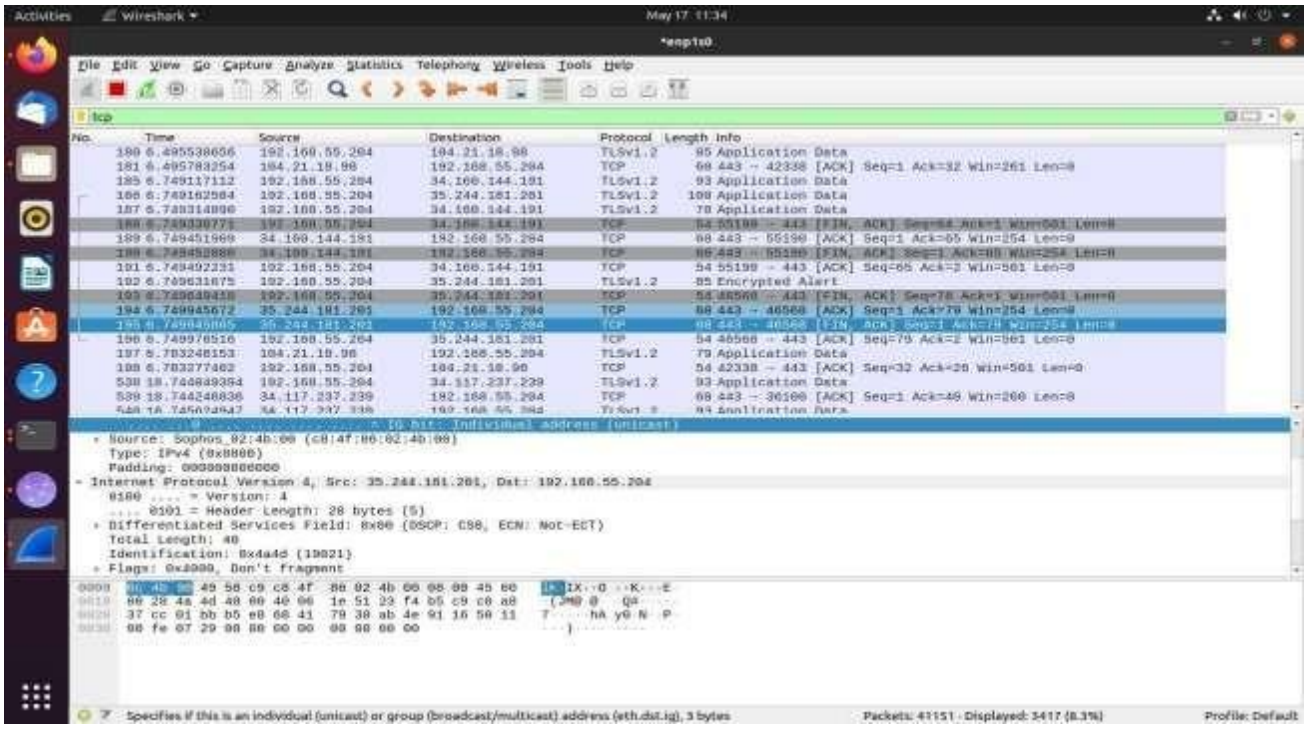
```
onOffHelper.SetAttribute ("OffTime", StringValue("ns3::ConstantRandomVariable[Constant=0]"));
    ApplicationContainer spokeApps;
    for (uint32_t i = 0; i < star.SpokeCount (); ++i)
      {
      AddressValue remoteAddress (InetSocketAddress
(star.GetHubIpv4Address (i), port));
          onOffHelper.SetAttribute ("Remote", remoteAddress); spokeApps.Add
          (onOffHelper.Install (star.GetSpokeNode (i)));
      }
    spokeApps.Start (Seconds (1.0));
    spokeApps.Stop (Seconds (10.0)); NS_LOG_INFO ("Enable
    static global routing.");
    //
    // Turn on global static routing so we can actually be routed across the star.
    //
    Ipv4GlobalRoutingHelper::PopulateRoutingTables ();NS_LOG_INFO
    ("Enable pcap tracing.");
    //
    // Do pcap tracing on all point-to-point devices on all nodes.
    //
    MobilityHelper mobility; mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
    mobility.Install(nodes);
    AnimationInterface anim("star.xml");
```

```
  AnimationInterface::SetConstantPosition(nodes.Get(0),10,2);
    AnimationInterface::SetConstantPosition(nodes.Get(1),11,5);
    AnimationInterface::SetConstantPosition(nodes.Get(2),15,2);
    AnimationInterface::SetConstantPosition(nodes.Get(3),19,7);
    anim.EnablePacketMetadata(true); pointToPoint.EnablePcapAll ("star");

    NS_LOG_INFO ("Run Simulation.");
    Simulator::Run (); Simulator::Destroy ();
    NS_LOG_INFO ("Done.");

    return 0;
}
```

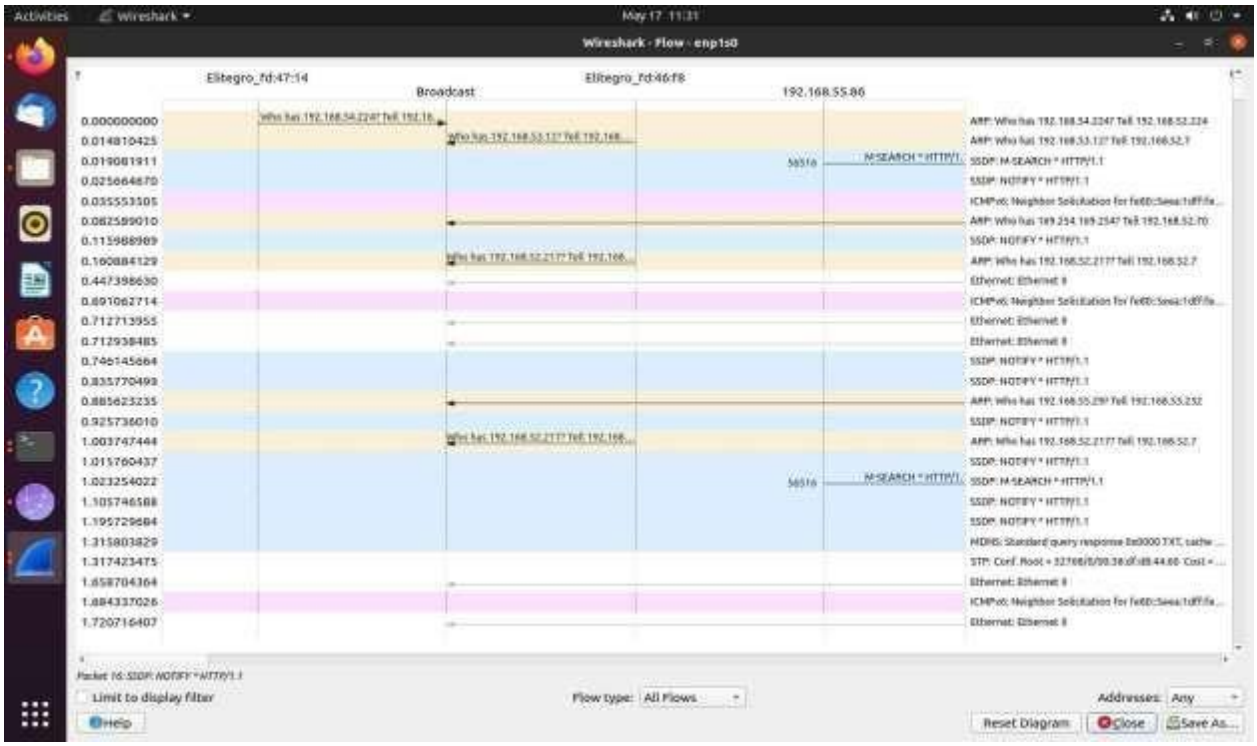
Output:

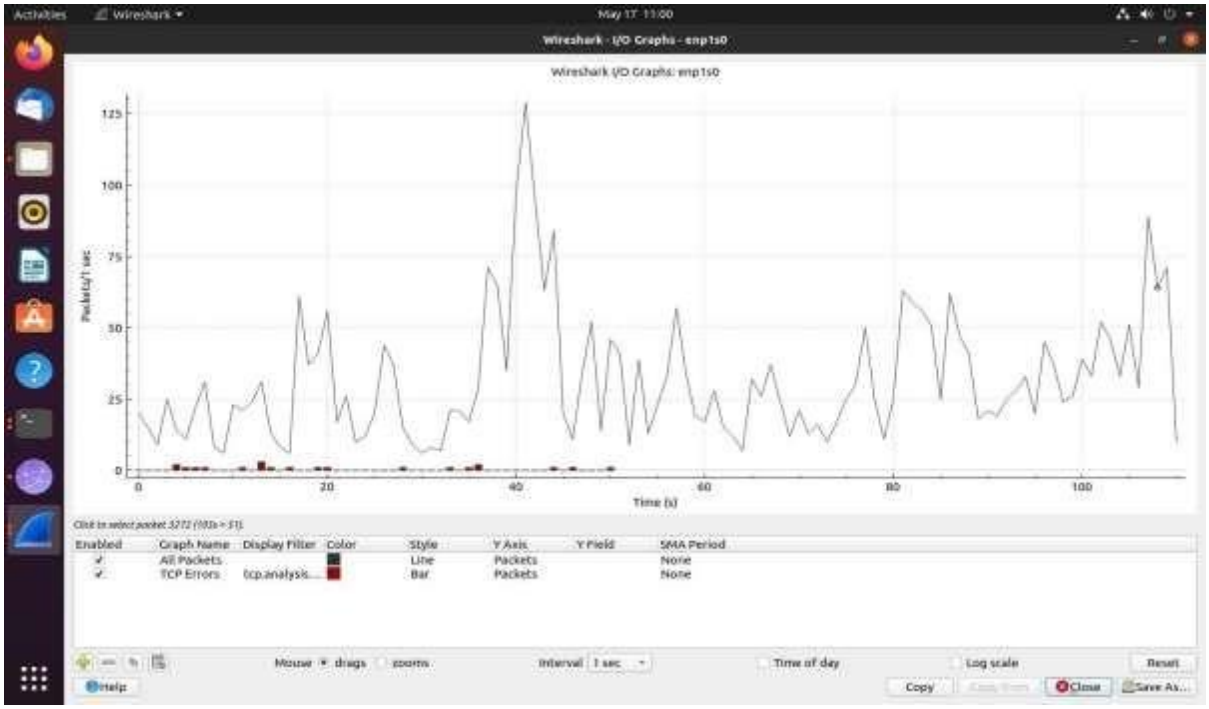


Roll No:-C22078
Name:-Pratik Vilas Mestry

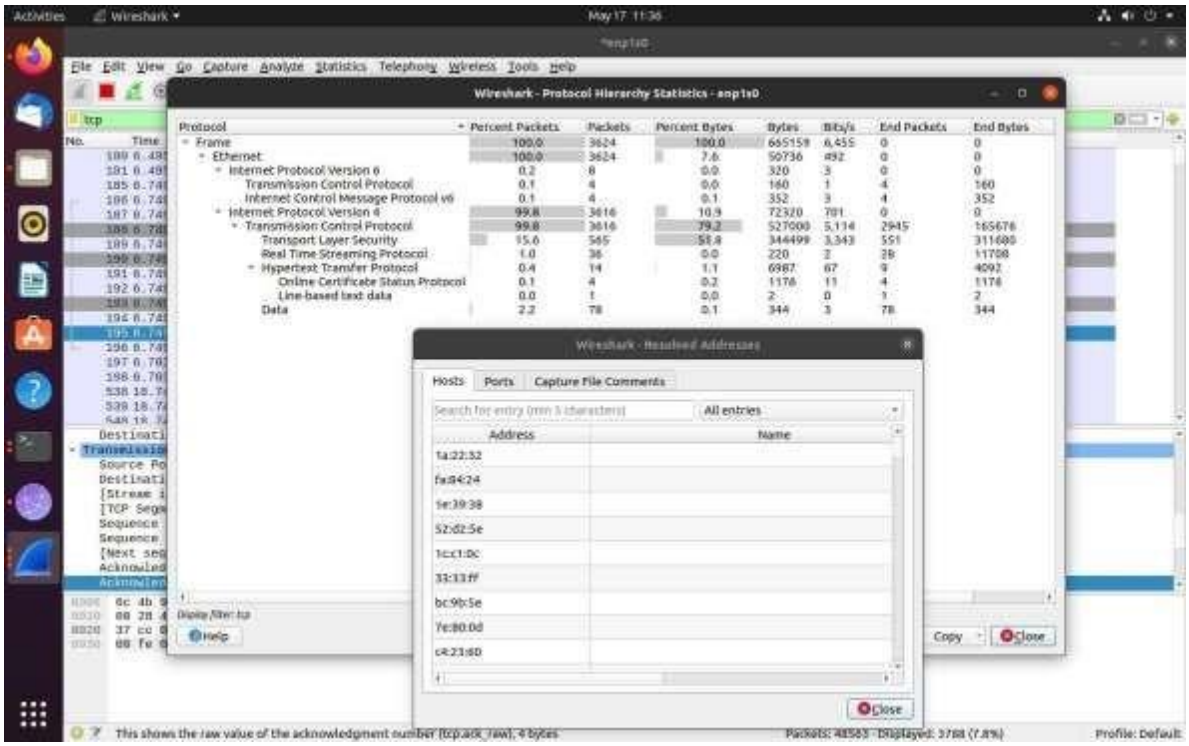


Roll No:-C22078
Name:-Pratik Vilas Mestry





Roll No:-C22078
Name:-Pratik Vilas Mestry



Roll No:-C22078
Name:-Pratik Vilas Mestry

Practical No 14

NETWORKING MINI PROJECT

Aim:- Animate 3 way hanshake for TCP connection using NetAnimator.

Tool Used:-

Three-Way HandShake or a TCP 3-way handshake is a process which is used in a TCP/IP network to make a connection between the server and client. It is a three-step process that requires both the client and server to exchange synchronization and acknowledgment packets before the real data communication process starts.

A **network simulator** is a software program that can predict the performance of a computer network or a wireless communication network. Since communication networks have become too complex for traditional analytical methods to provide an accurate understanding of system behavior, network simulators are used. In simulators, the computer network is modeled with devices, links, applications, etc., and the network performance is reported. Simulators come with support for the most popular technologies and networks in use today such as 5G, Internet of Things (IoT), Wireless LANs, mobile ad hoc networks, wireless sensor networks, vehicular ad hoc networks, cognitive radio networks, LTE etc.

There are a wide variety of network simulators, ranging from the very simple to the very complex. Minimally, a network simulator must enable a user to

- Model the network topology specifying the nodes on the network and the links between those nodes
- Model the application flow (traffic) between the nodes
- Providing network performance metrics as output
- Visualization of the packet flow
- Technology/protocol evaluation and device designs

- Logging of packet/events for drill-down analyses/debugging

Roll No:-C22078
Name:-Pratik Vilas Mestry

Source Code:-

```
// Default Network topology, 9 nodes in a star
/*
         n2 n3 n4 \ | /
           \|/
       n1---n0---n5
           /| \
          / | \
         n8 n7 n6
*/
// - CBR Traffic goes from the star "arms" to the "hub"
// - Tracing of queues and packet receptions to file
//   "tcp-star-server.tr"
// - pcap traces also generated in the following files
//   "tcp-star-server-$n-$i.pcap" where n and i represent node and interface
//   numbers respectively
// Usage examples for things you might want to tweak:
//       ./waf --run="tcp-star-server"
//       ./waf --run="tcp-star-server --nNodes=25"
//       ./waf --run="tcp-star-server --ns3::OnOffApplication::DataRate=10000"
//       ./waf --run="tcp-star-server --ns3::OnOffApplication::PacketSize=500"
// See the ns-3 tutorial for more info on the command line:
// http://www.nsnam.org/tutorials.html


#include <iostream>
#include <fstream>
#include <string>
#include <cassert>
#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"
#include "ns3/ipv4-global-routing-helper.h"

#include "ns3/netanim-module.h"
#include "ns3/mobility-module.h"
using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("TcpServer");
int
main (int argc, char *argv[])
{
  // Users may find it convenient to turn on explicit debugging
  // for selected modules; the below lines suggest how to do this
  //LogComponentEnable ("TcpServer", LOG_LEVEL_INFO);
  //LogComponentEnable ("TcpL4Protocol", LOG_LEVEL_ALL);
  //LogComponentEnable ("TcpSocketImpl", LOG_LEVEL_ALL);
  //LogComponentEnable ("PacketSink", LOG_LEVEL_ALL);
```

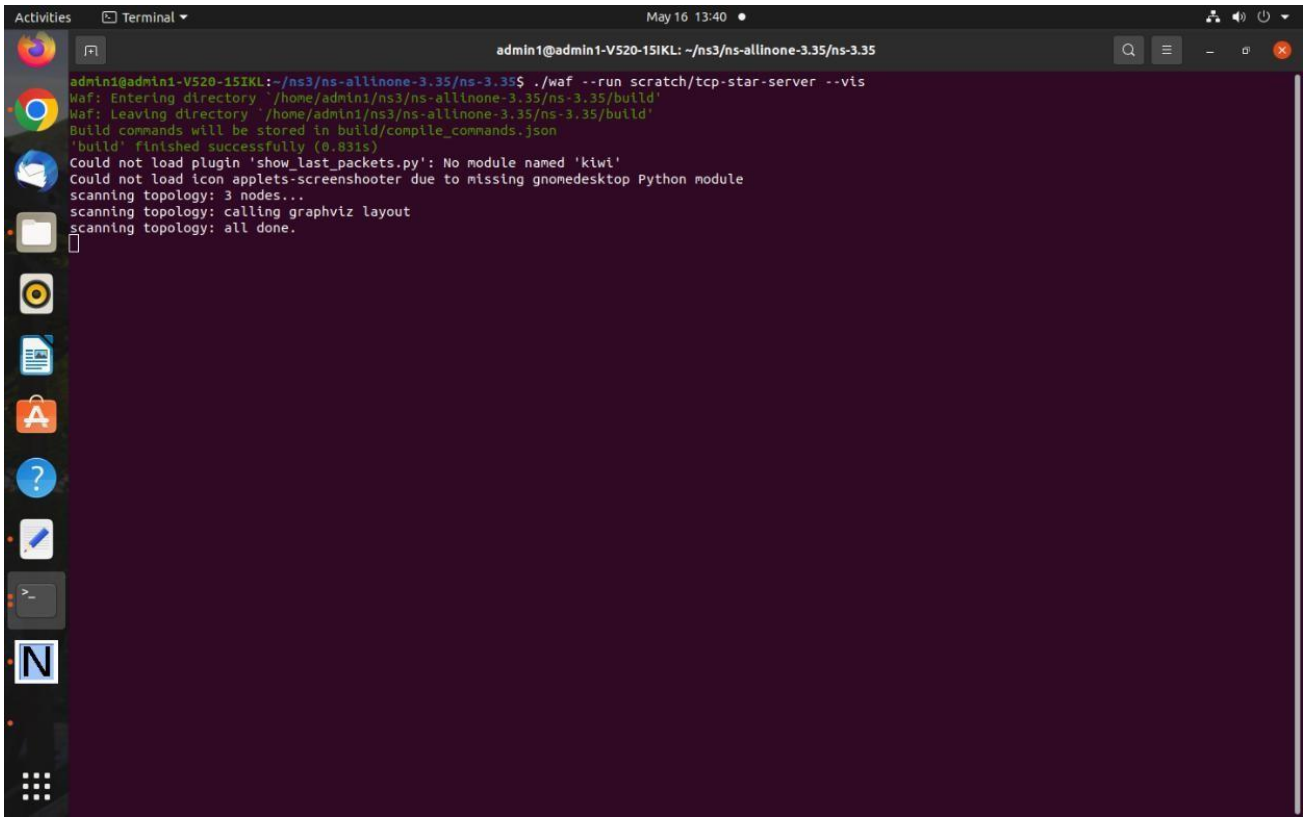
```
// Set up some default values for the simulation.
Config::SetDefault ("ns3::OnOffApplication::PacketSize", UintegerValue (250));
Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue ("5kb/s")); uint32_t N =
2; //number of nodes in the star
// Allow the user to override any of the defaults and the above
// Config::SetDefault()s at run-time, via command-line arguments
CommandLine cmd (__FILE__);
cmd.AddValue ("nNodes", "Number of nodes to place in the star", N); cmd.Parse
(argc, argv);
// Here, we will create N nodes in a star.
NS_LOG_INFO ("Create nodes.");
NodeContainer serverNode;
NodeContainer clientNode;
serverNode.Create (1);
clientNode.Create (N-1);
NodeContainer nodes = NodeContainer (serverNode, clientNode);
nodes.Create(1,1);

// Install network stacks on the nodes
InternetStackHelper internet;
internet.Install (nodes);


//Collect an adjacency list of nodes for the p2p topology
std::vector<NodeContainer> nodeAdjacencyList (N-1); for(uint32_t
i=0; i<nodeAdjacencyList.size (); ++i)
  {
    nodeAdjacencyList[i] = NodeContainer (serverNode, clientNode.Get (i));
  }

// We create the channels first without any IP addressing information
NS_LOG_INFO ("Create channels.");
PointToPointHelper p2p;
p2p.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
p2p.SetChannelAttribute ("Delay", StringValue ("2ms"));
std::vector<NetDeviceContainer> deviceAdjacencyList (N-1); for(uint32_t
i=0; i<deviceAdjacencyList.size (); ++i)
  {
    deviceAdjacencyList[i] = p2p.Install (nodeAdjacencyList[i]);
  }

// Later, we add IP addresses.
NS_LOG_INFO ("Assign IP Addresses.");
Ipv4AddressHelper ipv4;
std::vector<Ipv4InterfaceContainer> interfaceAdjacencyList (N-1);
for(uint32_t i=0; i<interfaceAdjacencyList.size (); ++i)
  {
    std::ostringstream subnet;
```

```
    subnet<<"10.1."<<i+1<<".0";
    ipv4.SetBase (subnet.str ().c_str (), "255.255.255.0"); interfaceAdjacencyList[i] =
    ipv4.Assign (deviceAdjacencyList[i]);
   }

  //Turn on global static routing
  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

  // Create a packet sink on the star "hub" to receive these packets uint16_t
  port = 50000;
  Address sinkLocalAddress (InetSocketAddress (Ipv4Address::GetAny (), port));
  PacketSinkHelper sinkHelper ("ns3::TcpSocketFactory", sinkLocalAddress);
  ApplicationContainer sinkApp = sinkHelper.Install (serverNode); sinkApp.Start (Seconds
  (1.0));
  sinkApp.Stop (Seconds (10.0));

  // Create the OnOff applications to send TCP to the server
OnOffHelper clientHelper ("ns3::TcpSocketFactory", Address ());

  clientHelper.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
  clientHelper.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));

  //normally wouldn't need a loop here but the server IP address is different
  //on each p2p subnet
  ApplicationContainer clientApps;
  for(uint32_t i=0; i<clientNode.GetN (); ++i)
    {
     AddressValue remoteAddress
       (InetSocketAddress (interfaceAdjacencyList[i].GetAddress (0), port));
     clientHelper.SetAttribute ("Remote", remoteAddress); clientApps.Add
     (clientHelper.Install (clientNode.Get (i)));
    }
  clientApps.Start (Seconds (1.0));
  clientApps.Stop (Seconds (10.0));

   MobilityHelper mobility;
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(nodes);

  AnimationInterface anim("tcp-star-server.xml");
  AnimationInterface::SetConstantPosition (nodes.Get(0),10,25);
  AnimationInterface::SetConstantPosition (nodes.Get(1),40,25);

anim.EnablePacketMetadata(true);
```

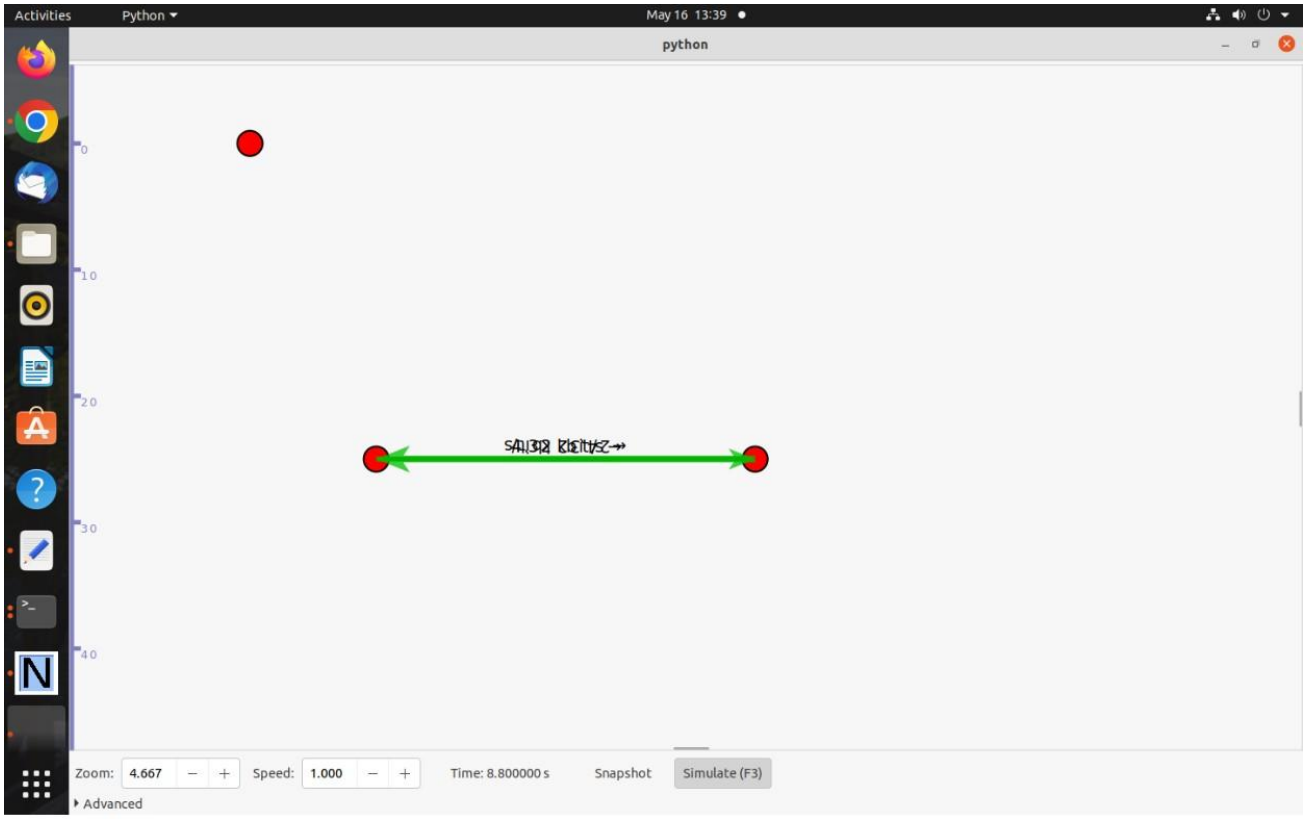
```
  //configure tracing
  AsciiTraceHelper ascii;
  p2p.EnableAsciiAll (ascii.CreateFileStream ("tcp-star-server.tr"));
  p2p.EnablePcapAll ("tcp-star-server");

  NS_LOG_INFO ("Run Simulation.");
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");

  return 0;
}
```

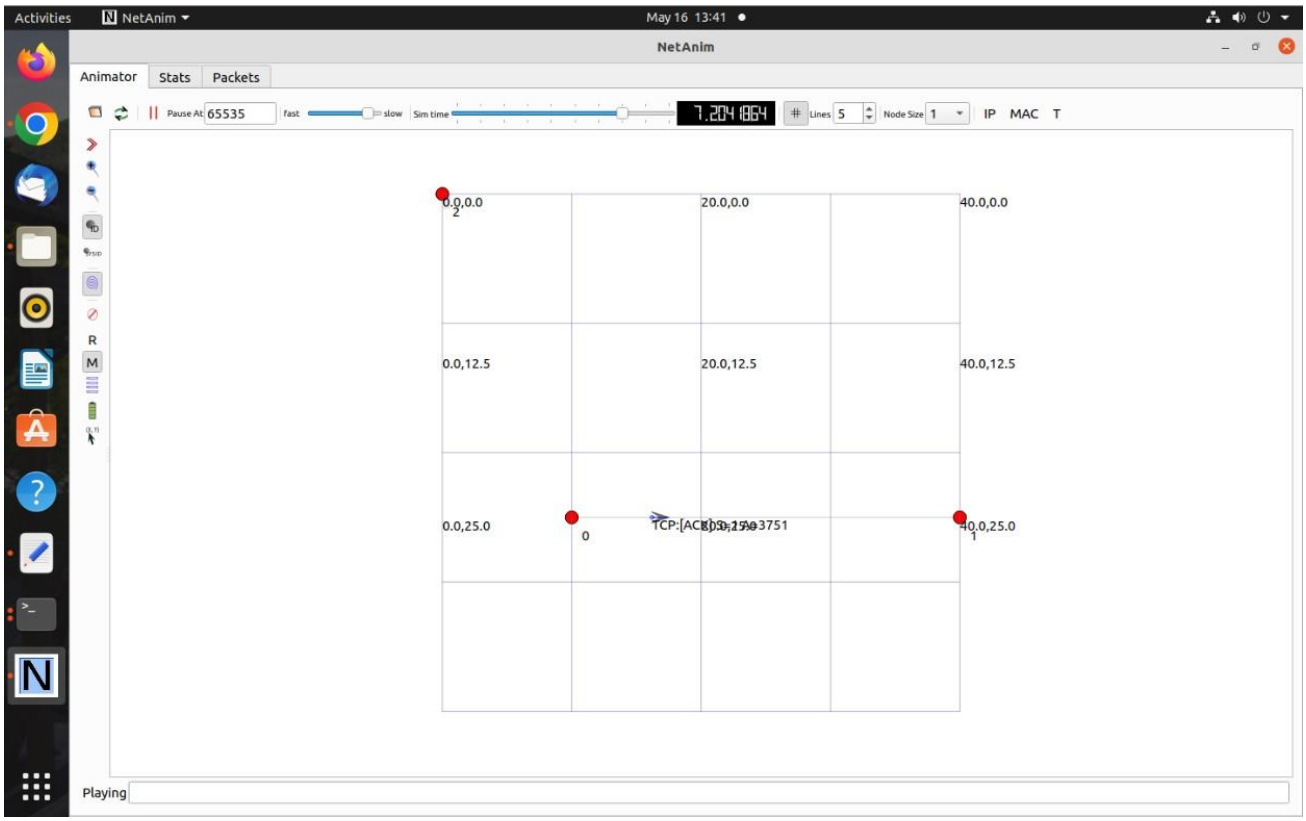
Roll No:-C22078
Name:-Pratik Vilas Mestry

Output:-

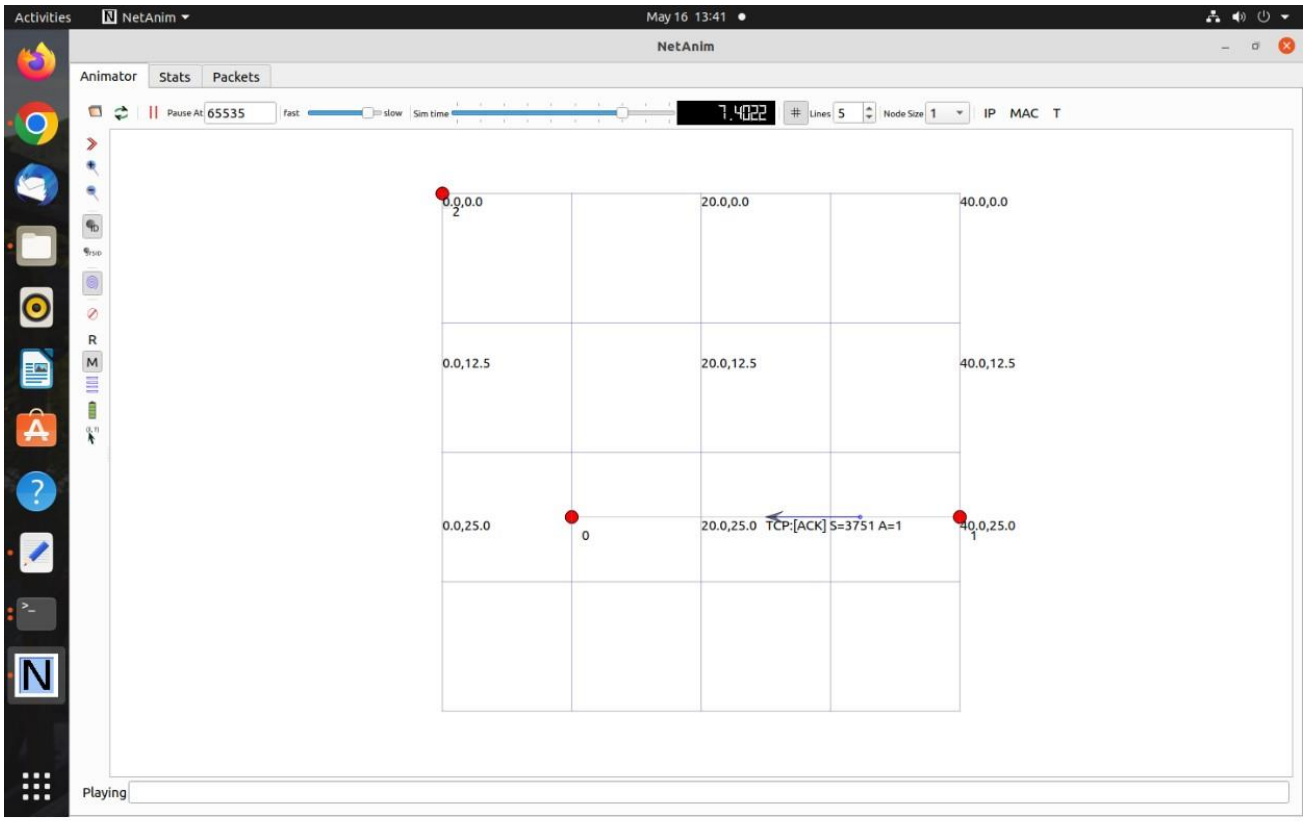


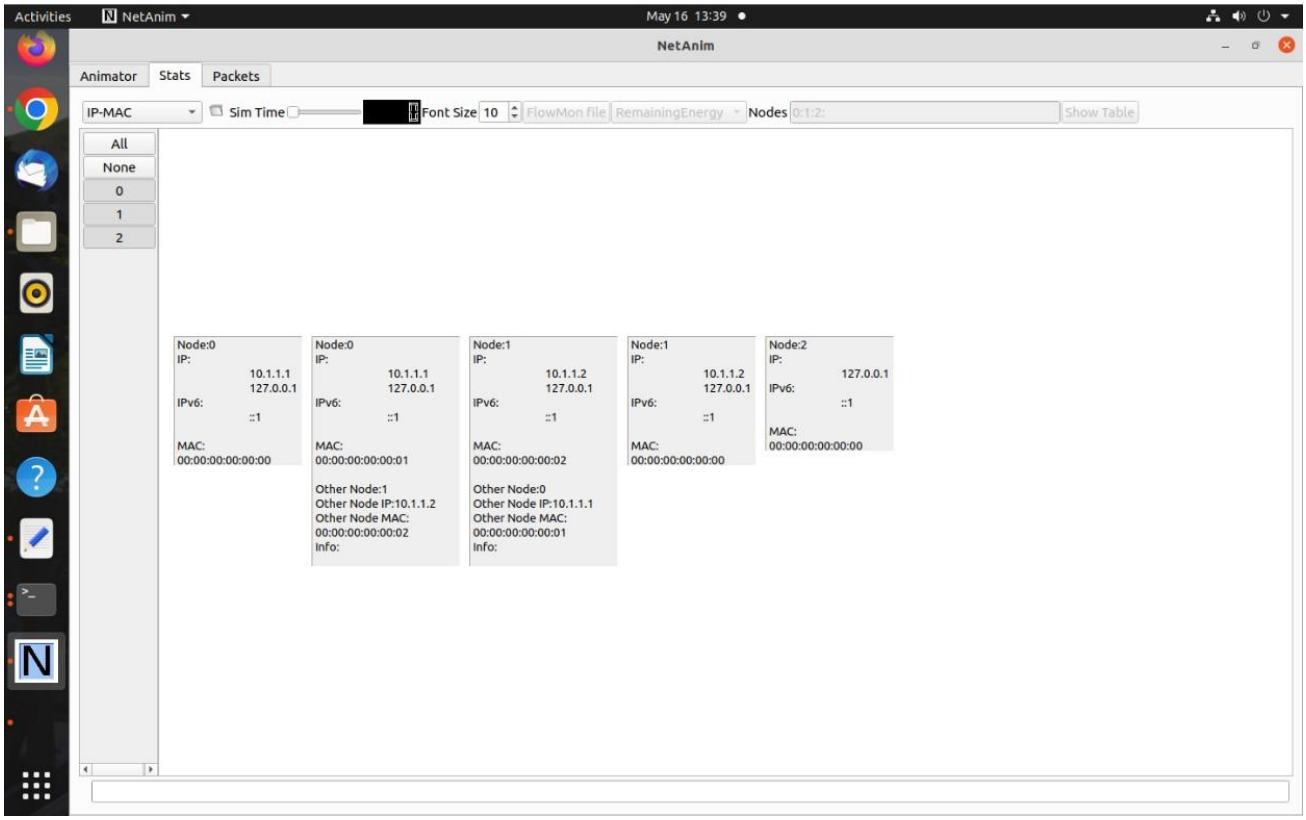
Roll No:-C22078
Name:-Pratik Vilas Mestry



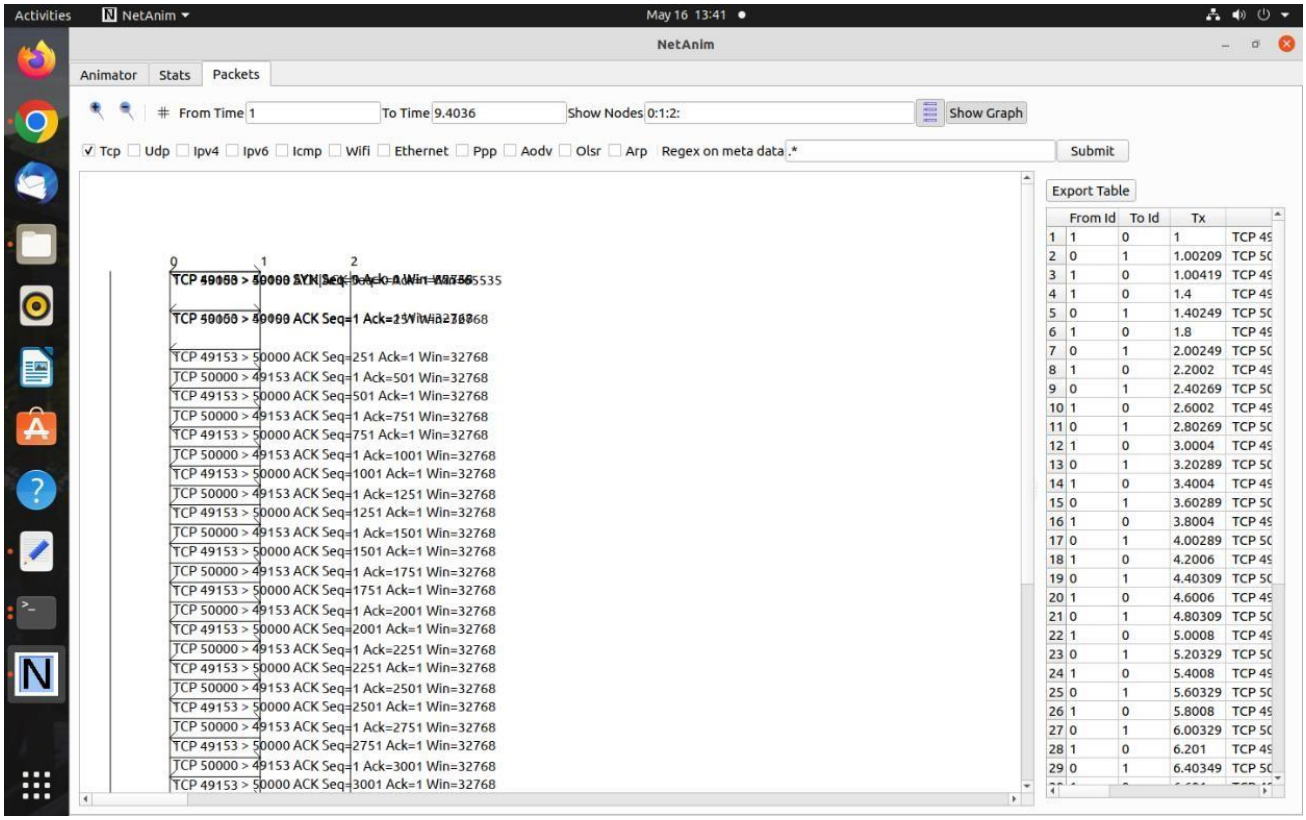


Roll No:-C22078
Name:-Pratik Vilas Mestry





Roll No:-C22078
Name:-Pratik Vilas Mestry



Conclusion:-

The three-way handshake is a method used by the Transmission Control Protocol (TCP) to establish a connection between two devices. The conclusion of the three-way handshake is the establishment of a reliable connection between the two devices, allowing them to exchange data.

The three-way handshake works as follows:

1.The initiating device (referred to as the client) sends a SYN (synchronize) packet to the receiving device (referred to as the server).

2.The server receives the SYN packet and responds with a SYN-ACK (synchronize-acknowledge) packet.

3.The client receives the SYN-ACK packet and responds with an ACK (acknowledge) packet.

At this point, both devices have exchanged a series of packets, and a reliable connection has been established. The client can now begin sending data to the server, and the server can acknowledge receipt of that data.

In summary, the conclusion of the three-way handshake using TCP connection is the establishment of a reliable connection between two devices, allowing them to exchange data.

Roll No:-C22078
Name:-Pratik Vilas Mestry