Practical - 2

Aim: Program to simulate Point to Point topology.

Point_89.cc

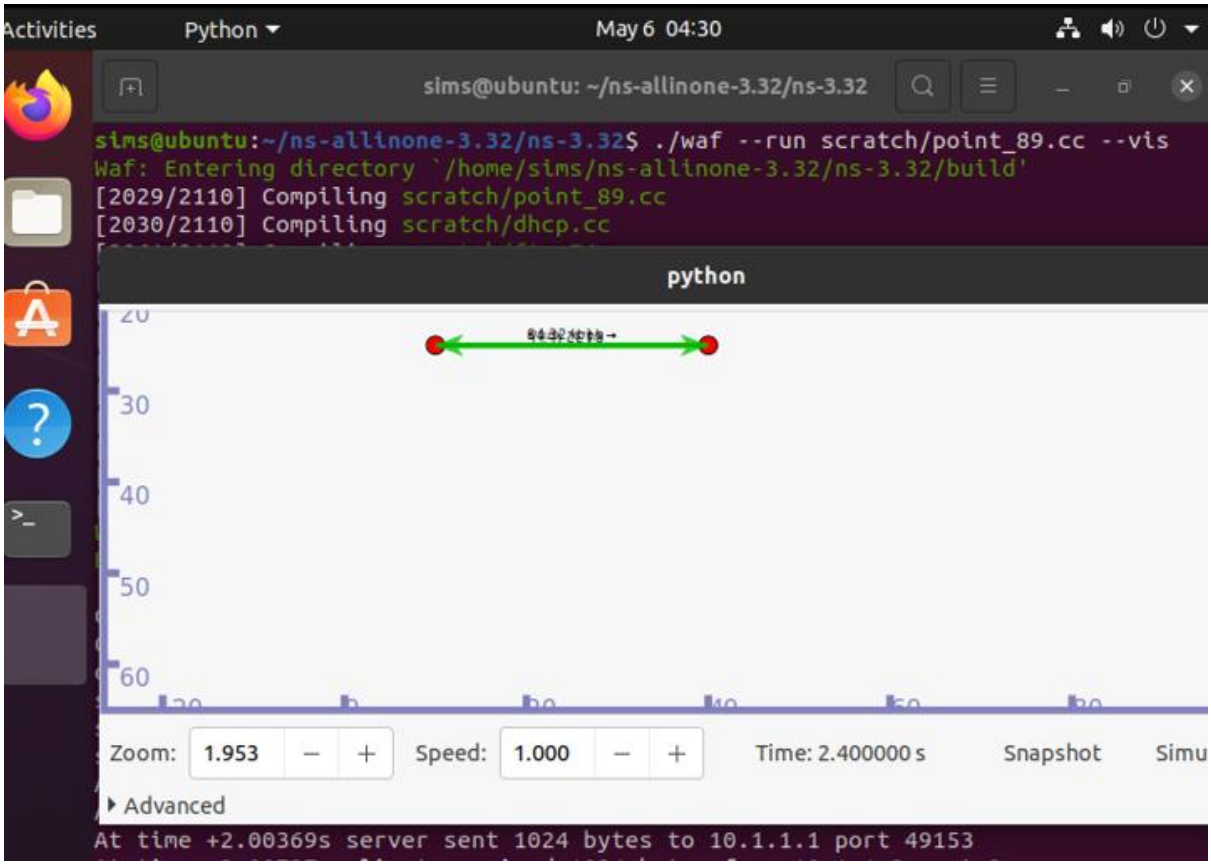
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/netanim-module.h"
#include "ns3/mobility-module.h"

using namespace ns3;
NS_LOG_COMPONENT_DEFINE("FirstScriptExample");
int main(int argc, char *argv[])
{
  // Parse command line arguments
  CommandLine cmd;
  cmd.Parse(argc, argv);

    Time::SetResolution(Time::NS);
  // Enable logging for the UDP Echo client and server applications
  LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);

  // Create two nodes (n0 and n1)
  NodeContainer nodes;
  nodes.Create(2);
```

```
// Set up the Point-to-Point link between the two nodes
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

// Install the Point-to-Point devices on the nodes
NetDeviceContainer devices;
devices = pointToPoint.Install(nodes);

// Install the Internet stack (IP, UDP, etc.) on the nodes
InternetStackHelper stack;
stack.Install(nodes);

// Set IP addresses for the devices
Ipv4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign(devices);

// Set up the UDP Echo server on node 1, listening on port 9
UdpEchoServerHelper echoServer(15);
ApplicationContainer serverApps = echoServer.Install(nodes.Get(1));
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));

// Set up the UDP Echo client on node 0, sending to node 1 at port 9
UdpEchoClientHelper echoClient(interfaces.GetAddress(1), 15);
echoClient.SetAttribute("MaxPackets", UintegerValue(1));
```

```
echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
echoClient.SetAttribute("PacketSize", UintegerValue(1024));

// Install the UDP Echo client application on node 0
ApplicationContainer clientApps = echoClient.Install(nodes.Get(0));
clientApps.Start(Seconds(2.0));
clientApps.Stop(Seconds(10.0));

// Set the mobility model for both nodes (Constant Position)
MobilityHelper mobility;
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(nodes);

// Set specific positions for the nodes (n0 at (10,25), n1 at (40,25))
AnimationInterface anim("first.xml");
AnimationInterface::SetConstantPosition(nodes.Get(0), 10, 25);
AnimationInterface::SetConstantPosition(nodes.Get(1), 40, 25);

// Enable packet metadata in NetAnim (optional but useful for debugging)
anim.EnablePacketMetadata(true);

// Enable Pcap trace for packet capture
pointToPoint.EnablePcapAll("point");

// Run the simulation
Simulator::Run();
Simulator::Destroy();
return 0;
```

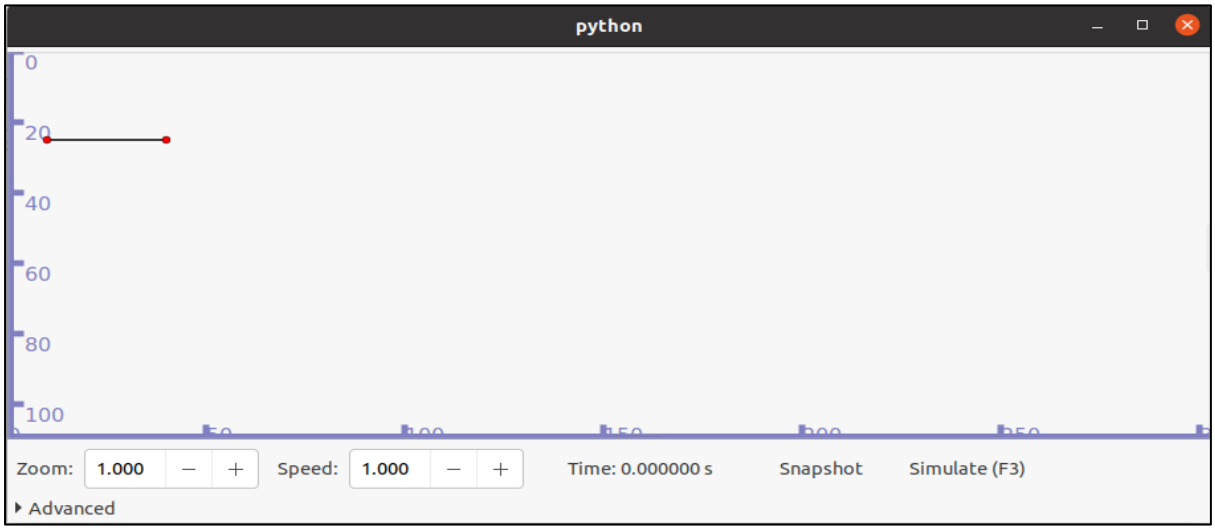
Output:

```
sims@ubuntu:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/point_89.cc --vis
Waf: Entering directory `/home/sims/ns-allinone-3.32/ns-3.32/build'
[2029/2110] Compiling scratch/point_89.cc
[2030/2110] Compiling scratch/dhcp.cc
[2061/2110] Compiling scratch/ftp_76.cc
[2062/2110] Linking build/scratch/dhcp
[2063/2110] Compiling scratch/mesh_76.cc
[2064/2110] Linking build/scratch/point_89
[2065/2110] Linking build/scratch/ftp_76
[2066/2110] Compiling scratch/mesh.cc
[2067/2110] Compiling scratch/hybrid_76.cc
[2068/2110] Linking build/scratch/mesh_76
[2069/2110] Linking build/scratch/mesh
[2070/2110] Linking build/scratch/hybrid_76
Waf: Leaving directory `/home/sims/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (9.304s)
```



Python Visualization:

```
sims@ubuntu:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/point_89 --vis
Waf: Entering directory `/home/sims/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/sims/ns-allinone-3.32/ns--3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.473s)
Could not load plugin 'show_last_packets.py': No module named 'kiwi'
Could not load icon applets-screenshooter due to missing gnomedesktop Python mod
ule
scanning topology: 2 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
```



2i. Point to point Topology using NetAnim

First_89.cc

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

// Default Network Topology
// 10.1.1.0
// n0 --- n1
// Point-to-Point link

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("FirstScriptExample");

int main(int argc, char *argv[])
{
   CommandLine cmd;
   cmd.Parse(argc, argv);
   // Set time resolution to nanoseconds
   Time::SetResolution(Time::NS);
```

```
// Enable logging for UDP Echo Client and Server
LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);

// Create 2 nodes
NodeContainer nodes;
nodes.Create(2);

// Configure point-to-point connection
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

// Install devices
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);

// Create and configure UDP Echo Server on node 1 (n1)
UdpEchoServerHelper echoServer(9);
ApplicationContainer serverApps = echoServer.Install(nodes.Get(1));
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));
```

```
// Create and configure UDP Echo Client on node 0 (n0)
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));

// Set constant mobility for animation
MobilityHelper mobility;
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(nodes);

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

// Enable PCAP tracing
pointToPoint.EnablePcapAll("pract2Mana");

// Run the simulation
Simulator::Run();
Simulator::Destroy();
return 0;
```

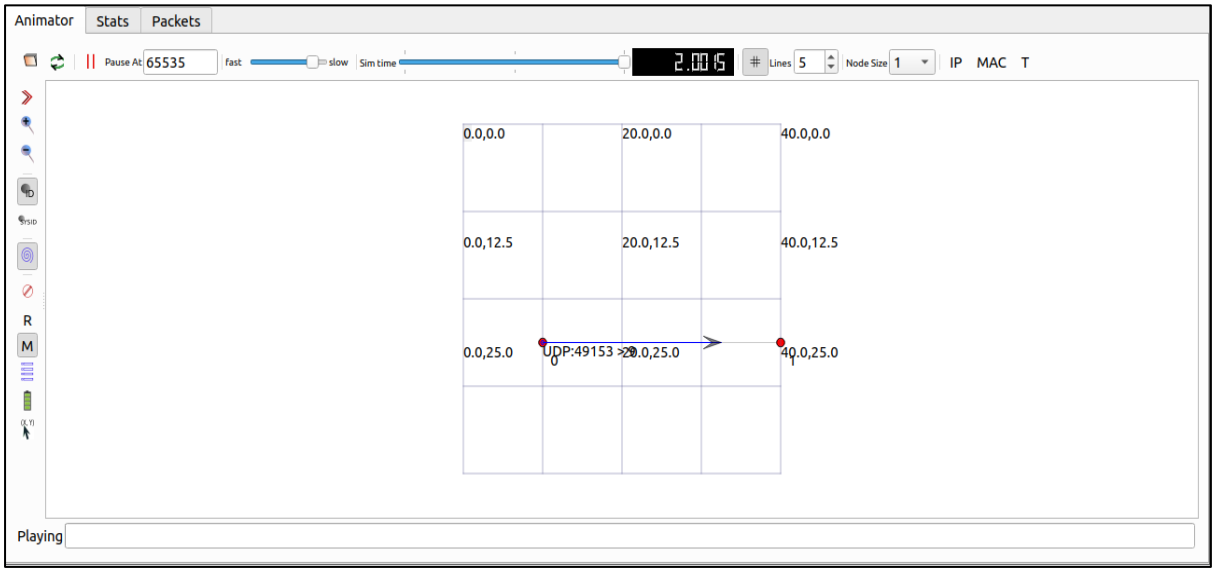
Output:

```
root36@ubuntu:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/first_89
Waf: Entering directory `/home/root36/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/root36/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.469s)
At time +2s client sent 1024 bytes to 10.1.1.2 port 9
At time +2.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time +2.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time +2.00737s client received 1024 bytes from 10.1.1.2 port 9
```

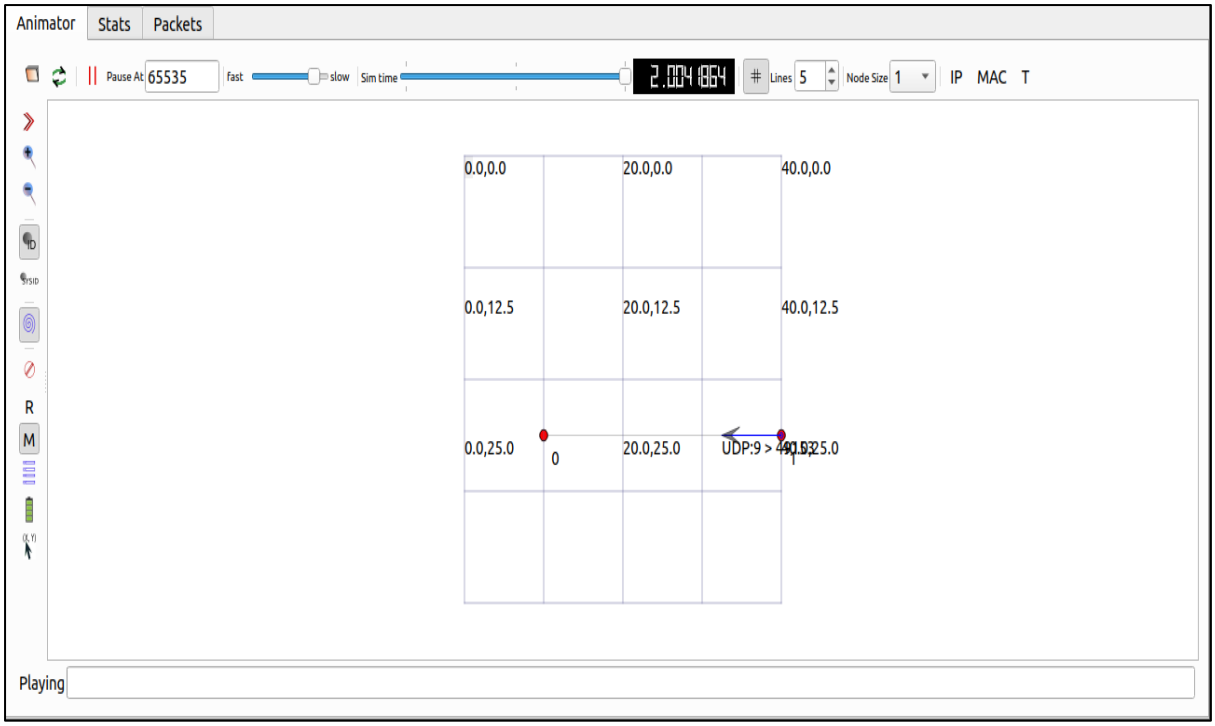
Open NetAnim to see the animation:

```
root36@ubuntu:~/ns-allinone-3.32/ns-3.32$ cd ~/ns-allinone-3.32/netanim-3.108/
root36@ubuntu:~/ns-allinone-3.32/netanim-3.108$ ./NetAnim
```

• Run the first.xml file for visualization:



Name:Bhargavikeni-89



Practical – 3

Aim: Program to simulate Bus topology

Code:

Bus_89.cc

```
#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"

using namespace ns3;
NS_LOG_COMPONENT_DEFINE("SecondScriptExample");
int main(int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 5;

  // Command line argument parsing
  CommandLine cmd;
  cmd.AddValue("nCsma", "Number of CSMA nodes/devices", nCsma);
  cmd.AddValue("verbose", "Enable logging if true", verbose);
  cmd.Parse(argc, argv);
  // Enable logging
  if (verbose)
  {
    LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
```

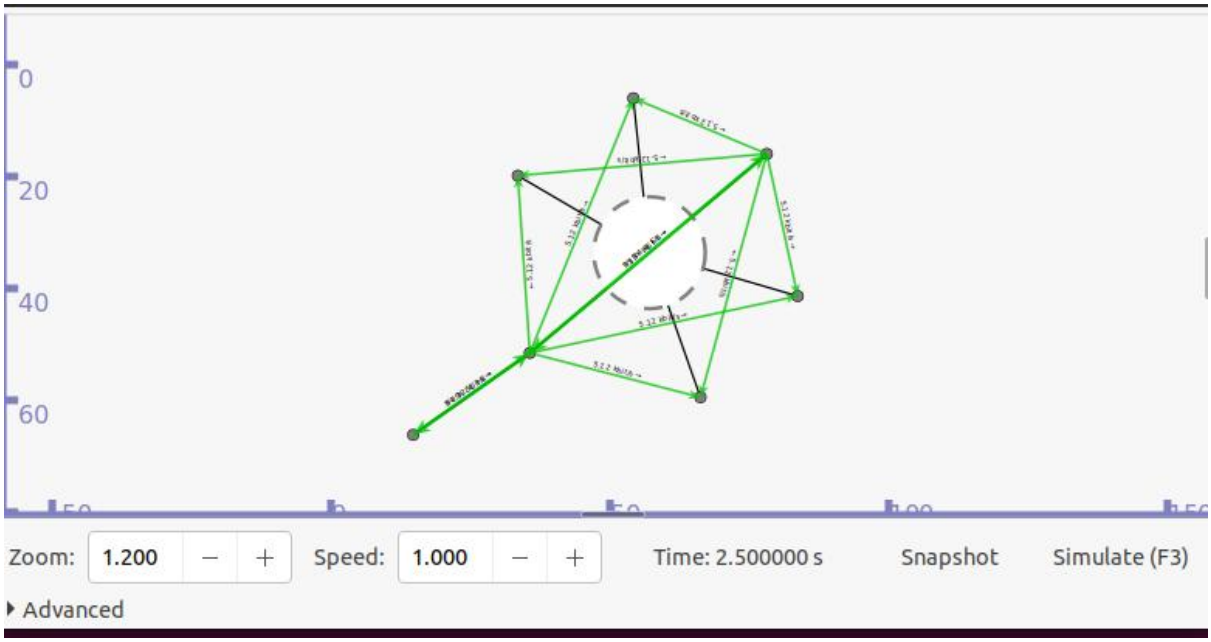
```
   LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
}
// Ensure at least 1 CSMA node
if (nCsma == 0)
{
   nCsma = 1;
}
// Create point-to-point nodes
NodeContainer p2pNodes;
p2pNodes.Create(2);
// Create CSMA nodes and connect one P2P node
NodeContainer csmaNodes;
csmaNodes.Add(p2pNodes.Get(1));
csmaNodes.Create(nCsma);

// Configure point-to-point link
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate", StringValue("10Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("5ms"));

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

// Configure CSMA network
CsmaHelper csma;
csma.SetChannelAttribute("DataRate", StringValue("100Mbps"));
csma.SetChannelAttribute("Delay", TimeValue(NanoSeconds(6560)));
```

```
NetDeviceContainer csmaDevices;
csmaDevices = csma.Install(csmaNodes);
// Install Internet stack
InternetStackHelper stack;
stack.Install(p2pNodes.Get(0));
stack.Install(csmaNodes);

// Assign IP addresses (P2P)
Ipv4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign(p2pDevices);

// Assign IP addresses (CSMA)
address.SetBase("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign(csmaDevices);

// Create UDP Echo Server (last CSMA node)
UdpEchoServerHelper echoServer(9);
ApplicationContainer serverApps;
serverApps = echoServer.Install(csmaNodes.Get(nCsma));
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));

// Create UDP Echo Client (P2P node)
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;
  clientApps = echoClient.Install(p2pNodes.Get(0));
  clientApps.Start(Seconds(2.0));
  clientApps.Stop(Seconds(10.0));
  // Populate routing tables
  Ipv4GlobalRoutingHelper::PopulateRoutingTables();

  // Enable PCAP tracing
  pointToPoint.EnablePcapAll("pract2Mana");
  csma.EnablePcap("pract2Mana", csmaDevices.Get(1), true);
  // Run simulation
  Simulator::Run();
  Simulator::Destroy();
  return 0;
}
```

OUTPUT:

```
sims@ubuntu:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/bus_89.cc --vis
Waf: Entering directory `/home/sims/ns-allinone-3.32/ns-3.32/build'
[1995/2112] Compiling scratch/bus_89.cc
[1996/2112] Compiling scratch/bus.cc
[1997/2112] Compiling scratch/hybrid.cc
[1998/2112] Compiling scratch/ftp_72.cc
[1999/2112] Linking build/scratch/bus_89
[2000/2112] Linking build/scratch/bus
[2001/2112] Linking build/scratch/ftp_72
[2002/2112] Compiling scratch/first.cc
[2003/2112] Compiling scratch/dhcp_76.cc
[2004/2112] Compiling scratch/hybrid_72.cc
[2005/2112] Linking build/scratch/hybrid
[2006/2112] Linking build/scratch/first
[2007/2112] Linking build/scratch/dhcp_76
[2008/2112] Compiling scratch/udp_72.cc
[2009/2112] Compiling scratch/star.cc
[2010/2112] Linking build/scratch/hybrid_72
[2011/2112] Linking build/scratch/udp_72
[2012/2112] Compiling scratch/udp.cc
[2013/2112] Compiling scratch/mesh_72.cc
[2014/2112] Linking build/scratch/star
[2015/2112] Linking build/scratch/udp
[2016/2112] Compiling scratch/subdir/scratch-simulator-subdir.cc
```



Practical – 4

Aim: Program to simulate Start topology

Code:

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

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("StarExample");

int main(int argc, char *argv[])
{
  // Set default packet size and data rate
  Config::SetDefault("ns3::OnOffApplication::PacketSize", UintegerValue(137));
  Config::SetDefault("ns3::OnOffApplication::DataRate", StringValue("14kb/s"));

  // Default number of spokes
  uint32_t nSpokes = 8;
```

```
// Command line argument
CommandLine cmd;
cmd.AddValue("nSpokes", "Number of nodes in the star", nSpokes);
cmd.Parse(argc, argv);

// Enable logging
LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);

NS_LOG_INFO("Build star topology");

// Configure point-to-point links
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

// Create star topology
PointToPointStarHelper star(nSpokes, pointToPoint);
NS_LOG_INFO("Install Internet Stack");
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 Packet Sink (Server) on Hub
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 on each spoke
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 Routing");
Ipv4GlobalRoutingHelper::PopulateRoutingTables();

NS_LOG_INFO("Enable PCAP tracing");
pointToPoint.EnablePcapAll("star");
```

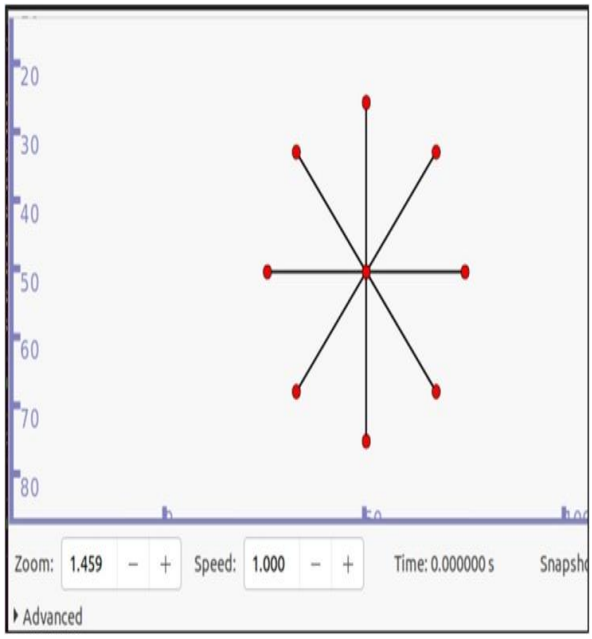
```
    // Set layout for NetAnim
    star.BoundingBox(1, 1, 100, 100);

    // NetAnim output file
    AnimationInterface anim("star.xml");

    NS_LOG_INFO("Run Simulation");
    Simulator::Run();
    Simulator::Destroy();
    NS_LOG_INFO("Done");
    return 0;
}
```

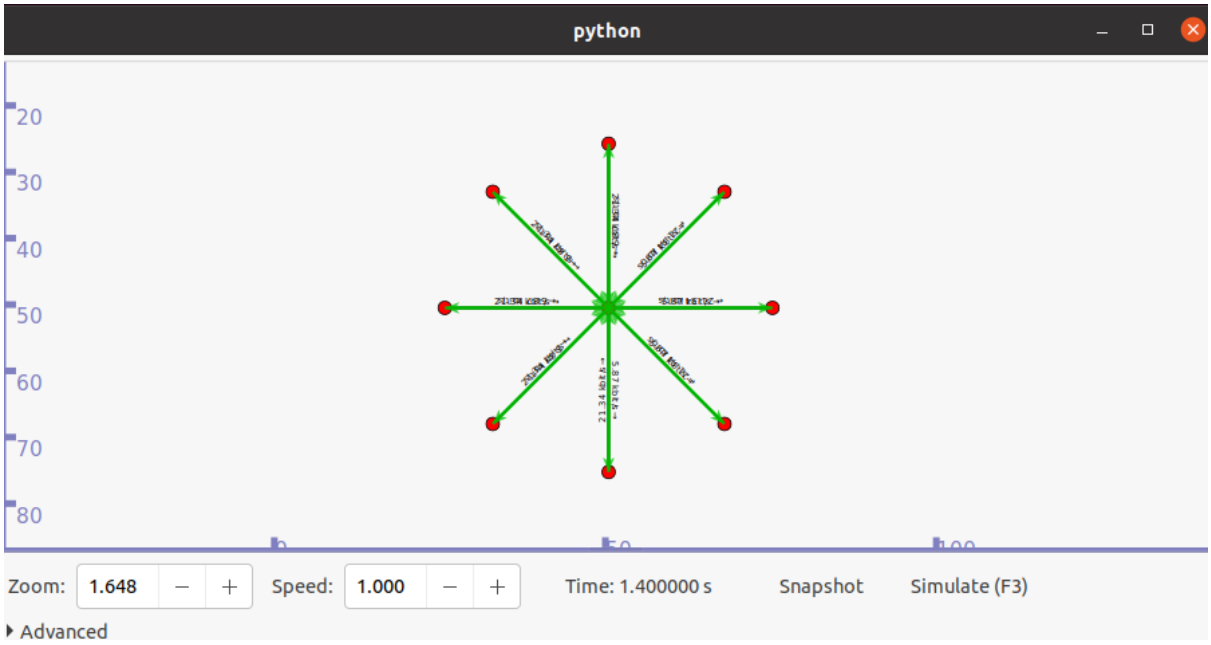
Output:

```
sims@ubuntu:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/star_89.cc --vis
Waf: Entering directory `/home/sims/ns-allinone-3.32/ns-3.32/build'
[2011/2114] Compiling scratch/star_89.cc
[2012/2114] Compiling scratch/star.cc
[2013/2114] Compiling scratch/udp.cc
[2014/2114] Linking build/scratch/star
[2015/2114] Compiling scratch/mesh_72.cc
[2016/2114] Linking build/scratch/star_89
[2017/2114] Linking build/scratch/udp
[2018/2114] Compiling scratch/subdir/scratch-simulator-subdir.cc
[2019/2114] Compiling scratch/star_76.cc
[2020/2114] Linking build/scratch/subdir/subdir
[2021/2114] Compiling scratch/bus_76.cc
[2022/2114] Linking build/scratch/mesh_72
[2023/2114] Linking build/scratch/star_76
[2024/2114] Linking build/scratch/bus_76
[2025/2114] Compiling scratch/ftp.cc
[2056/2114] Compiling scratch/scratch-simulator.cc
[2057/2114] Compiling scratch/point_76.cc
[2058/2114] Linking build/scratch/scratch-simulator
[2059/2114] Linking build/scratch/ftp
```

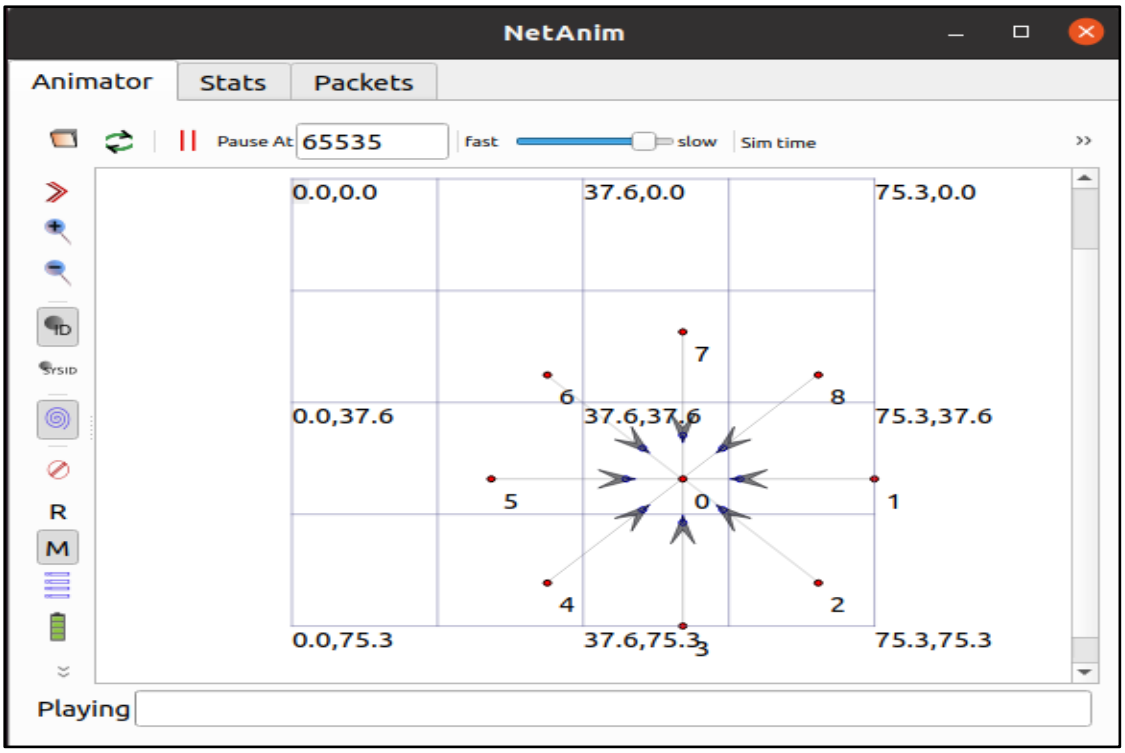
Name:Bhargavikeni-89

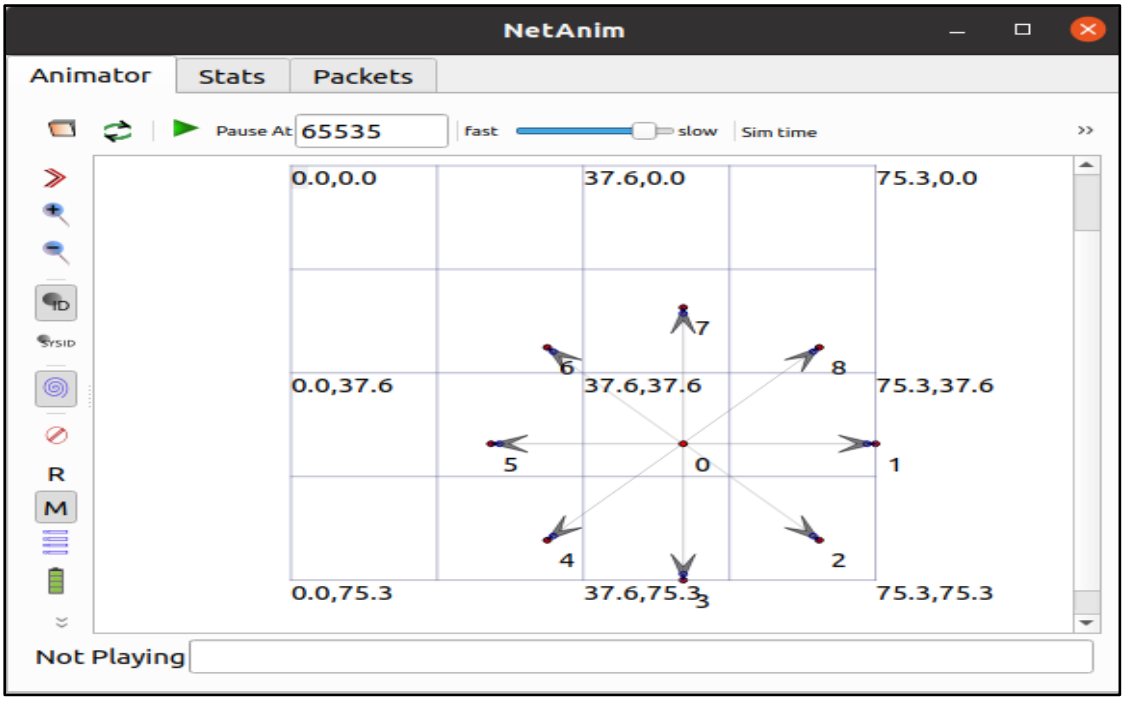
Using Python Visualizer :





Name:Bhargavikeni-89





Practical – 5

Aim: Program to simulate Mesh topology

Code:

```
#include "ns3/applications-module.h"
#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/mesh-helper.h"
#include "ns3/mesh-module.h"
#include "ns3/mobility-module.h"
#include "ns3/network-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/netanim-module.h"

#include <fstream>
#include <iostream>
#include <sstream>

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("MeshExample");
// Global counters
uint32_t g_udpTxCount = 0;
uint32_t g_udpRxCount = 0;
// TX Trace
void TxTrace(Ptr<const Packet> p)
```

```
{
  NS_LOG_DEBUG("Sent " << p->GetSize() << " bytes");
  g_udpTxCount++;
}
// RX Trace
void RxTrace(Ptr<const Packet> p)
{
  NS_LOG_DEBUG("Received " << p->GetSize() << " bytes");
  g_udpRxCount++;
}
class MeshTest
{
public:
  MeshTest();
  void Configure(int argc, char **argv);
  int Run();

private:
  // Parameters
  int m_xSize;
  int m_ySize;
  double m_step;
  double m_randomStart;
  double m_totalTime;
  double m_packetInterval;
  uint16_t m_packetSize;
  uint32_t m_nIfaces;
  bool m_chan;
```

```
    bool m_pcap;
    bool m_ascii;
    std::string m_stack;
    std::string m_root;
    // Containers
    NodeContainer nodes;
    NetDeviceContainer meshDevices;
    Ipv4InterfaceContainer interfaces;
    MeshHelper mesh;
    // Functions
    void CreateNodes();
    void InstallInternetStack();
    void InstallApplication();
    void Report();
};

MeshTest::MeshTest()
    : m_xSize(3),
      m_ySize(3),
      m_step(50.0),
      m_randomStart(0.1),
      m_totalTime(100.0),
      m_packetInterval(1.0),
      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;
  cmd.AddValue("x-size", "Grid width", m_xSize);
  cmd.AddValue("y-size", "Grid height", m_ySize);
  cmd.AddValue("step", "Distance between nodes", m_step);
  cmd.AddValue("start", "Random start delay", m_randomStart);
  cmd.AddValue("time", "Simulation time", m_totalTime);
  cmd.AddValue("packet-interval", "Packet interval", m_packetInterval);
  cmd.AddValue("packet-size", "Packet size", m_packetSize);
  cmd.AddValue("interfaces", "Number of interfaces", m_nIfaces);
  cmd.AddValue("channels", "Use multiple channels", m_chan);
  cmd.AddValue("pcap", "Enable PCAP", m_pcap);
  cmd.AddValue("ascii", "Enable ASCII", m_ascii);
  cmd.AddValue("stack", "Protocol stack", m_stack);
  cmd.AddValue("root", "Root MAC", m_root);
  cmd.Parse(argc, argv);

  if (m_ascii)
  {
    PacketMetadata::Enable();
  }
}
//
```

```
void MeshTest::CreateNodes()
{
  nodes.Create(m_xSize * m_ySize);
  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default();
  YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default();
  wifiPhy.SetChannel(wifiChannel.Create());
  mesh = MeshHelper::Default();
  if (!Mac48Address(m_root.c_str()).IsBroadcast())
  {
    mesh.SetStackInstaller(m_stack, "Root",
      Mac48AddressValue(Mac48Address(m_root.c_str())));
  }
  else
  {
    mesh.SetStackInstaller(m_stack);
  }
  mesh.SetSpreadInterfaceChannels(
    m_chan ? MeshHelper::SPREAD_CHANNELS : MeshHelper::ZERO_CHANNEL);

  mesh.SetMacType("RandomStart", TimeValue(Seconds(m_randomStart)));
  mesh.SetNumberOfInterfaces(m_nIfaces);

  meshDevices = mesh.Install(wifiPhy, nodes);
  // Mobility
  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("mesh");
  }
}
void MeshTest::InstallInternetStack()
{
  InternetStackHelper internet;
  internet.Install(nodes);

  Ipv4AddressHelper address;
  address.SetBase("10.1.1.0", "255.255.255.0");
  interfaces = address.Assign(meshDevices);
}
//
void MeshTest::InstallApplication()
{
  uint16_t port = 9;
  uint32_t sinkNode = m_xSize * m_ySize - 1;
  UdpEchoServerHelper server(port);
  ApplicationContainer serverApp = server.Install(nodes.Get(sinkNode));
  serverApp.Start(Seconds(1.0));
```

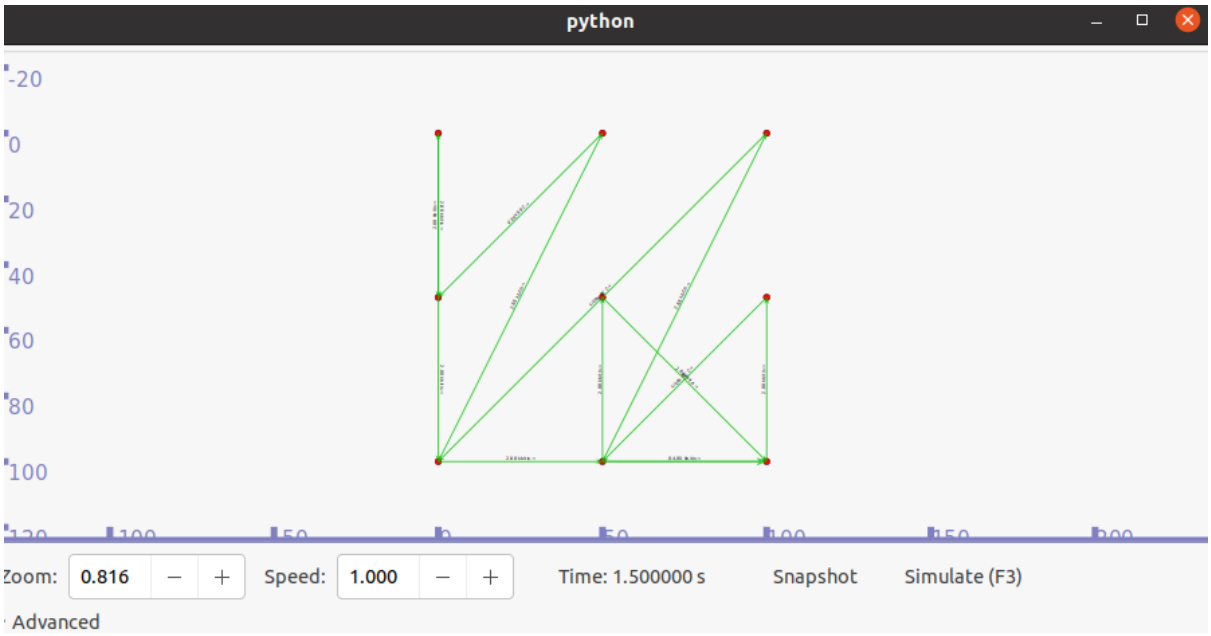
```
  serverApp.Stop(Seconds(m_totalTime + 1));


  UdpEchoClientHelper client(interfaces.GetAddress(sinkNode), port);
  client.SetAttribute("MaxPackets",
    UintegerValue((uint32_t)(m_totalTime / m_packetInterval)));
  client.SetAttribute("Interval",
    TimeValue(Seconds(m_packetInterval)));
  client.SetAttribute("PacketSize",
    UintegerValue(m_packetSize));
  ApplicationContainer clientApp = client.Install(nodes.Get(0));
  Ptr<UdpEchoClient> app = clientApp.Get(0)->GetObject<UdpEchoClient>();
  app->TraceConnectWithoutContext("Tx", MakeCallback(&TxTrace));
  app->TraceConnectWithoutContext("Rx", MakeCallback(&RxTrace));
  clientApp.Start(Seconds(1.0));
  clientApp.Stop(Seconds(m_totalTime + 1.5));
}
int MeshTest::Run()
{
  CreateNodes();
  InstallInternetStack();
  InstallApplication();
  AnimationInterface anim("mesh-topology.xml");
  Simulator::Schedule(Seconds(m_totalTime), &MeshTest::Report, this);
  Simulator::Stop(Seconds(m_totalTime + 2));
  Simulator::Run();
  Simulator::Destroy();
  std::cout << "Packets Sent: " << g_udpTxCount
        << " Received: " << g_udpRxCount << std::endl;
```

```
  return 0;
}
void MeshTest::Report()
{
  unsigned n = 0;

  for (auto i = meshDevices.Begin(); i != meshDevices.End(); ++i, ++n)
  {
    std::ostringstream os;
    os << "mp-report-" << n << ".xml";

    std::ofstream of(os.str());
    mesh.Report(*i, of);
    of.close();
  }
}
int main(int argc, char *argv[])
{
  MeshTest t;
  t.Configure(argc, argv);
  return t.Run();
}
```

Output:

```
sims@ubuntu:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/mesh_89.cc --vis
Waf: Entering directory `/home/sims/ns-allinone-3.32/ns-3.32/build'
[2039/2116] Compiling scratch/mesh_89.cc
[2040/2116] Compiling scratch/mesh_76.cc
[2071/2116] Compiling scratch/mesh.cc
[2072/2116] Linking build/scratch/mesh_76
[2073/2116] Compiling scratch/hybrid_76.cc
[2074/2116] Linking build/scratch/mesh_89
[2075/2116] Linking build/scratch/hybrid_76
[2076/2116] Linking build/scratch/mesh
Waf: Leaving directory `/home/sims/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (7.908s)
Could not load plugin 'show_last_packets.py': No module named 'kiwi'
Could not load icon applets-screenshooter due to missing gnomedesktop Python module
scanning topology: 9 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
```

Using python Visualizer:



Practical - 6

Aim: Program to simulate Hybrid topology

Code:

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/ssid.h"
#include "ns3/netanim-module.h"
#include "ns3/flow-monitor-helper.h"

using namespace ns3;
NS_LOG_COMPONENT_DEFINE("ThirdScriptExample");
int main(int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 3;
  uint32_t nWifi = 3;
  bool tracing = false;
  CommandLine cmd(__FILE__);
  cmd.AddValue("nCsma", "Number of CSMA nodes", nCsma);
  cmd.AddValue("nWifi", "Number of WiFi STA nodes", nWifi);
  cmd.AddValue("verbose", "Enable logging", verbose);
  cmd.AddValue("tracing", "Enable PCAP tracing", tracing);
```

```
cmd.Parse(argc, argv);
if (nWifi > 18)
{
   std::cout << "nWifi should be <= 18 (layout limit)" << std::endl;
   return 1;
}
if (verbose)
{
   LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
   LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
}
// ================= P2P SETUP =================
NodeContainer p2pNodes;
p2pNodes.Create(2);

PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));
NetDeviceContainer p2pDevices = pointToPoint.Install(p2pNodes);
// ================= CSMA SETUP =================
NodeContainer csmaNodes;
csmaNodes.Add(p2pNodes.Get(1));
csmaNodes.Create(nCsma);
CsmaHelper csma;
csma.SetChannelAttribute("DataRate", StringValue("100Mbps"));
csma.SetChannelAttribute("Delay", TimeValue(NanoSeconds(6560)));
NetDeviceContainer csmaDevices = csma.Install(csmaNodes);
// ================= WIFI SETUP =================
```

```
NodeContainer wifiStaNodes;
wifiStaNodes.Create(nWifi);
NodeContainer wifiApNode = p2pNodes.Get(0);
YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default();
phy.SetChannel(channel.Create());
WifiHelper wifi;
wifi.SetRemoteStationManager("ns3::AarfWifiManager");

WifiMacHelper mac;
Ssid ssid = Ssid("ns-3-ssid");
// STA
mac.SetType("ns3::StaWifiMac",
        "Ssid", SsidValue(ssid),
        "ActiveProbing", BooleanValue(false));
NetDeviceContainer staDevices = wifi.Install(phy, mac, wifiStaNodes);
// AP
mac.SetType("ns3::ApWifiMac",
        "Ssid", SsidValue(ssid));
NetDeviceContainer apDevices = wifi.Install(phy, mac, wifiApNode);
// ================= MOBILITY =================
MobilityHelper mobility;
mobility.SetPositionAllocator("ns3::GridPositionAllocator",
                        "MinX", DoubleValue(0.0),
                        "MinY", DoubleValue(0.0),
                        "DeltaX", DoubleValue(5.0),
                        "DeltaY", DoubleValue(10.0),
                        "GridWidth", UintegerValue(3),
```

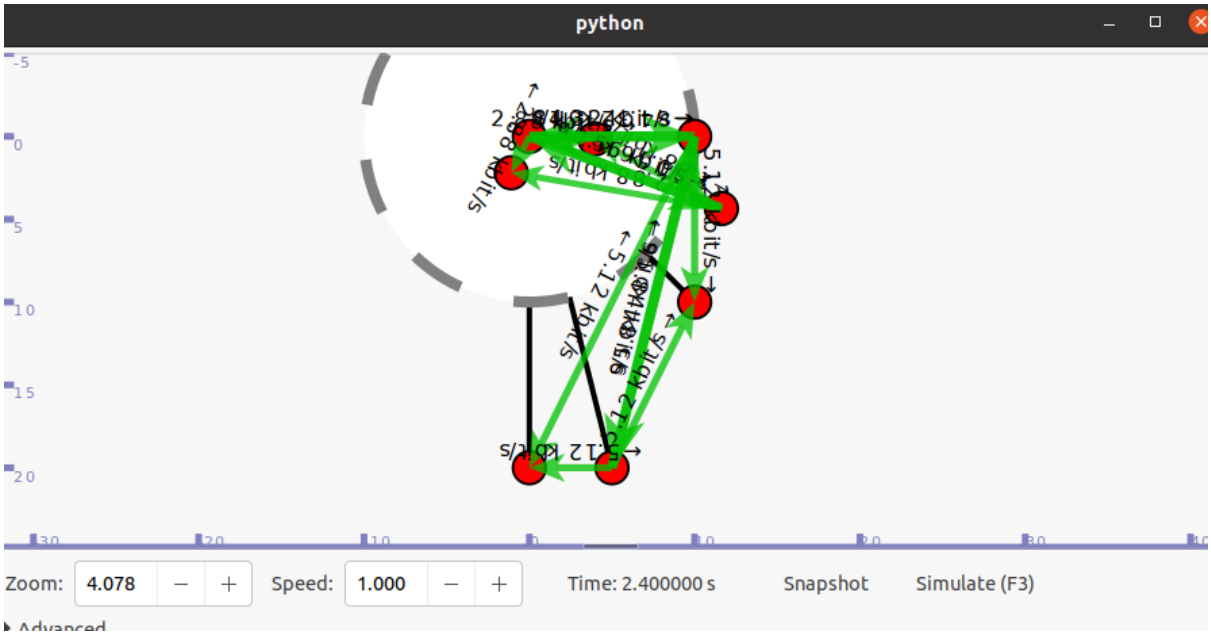
```
                    "LayoutType", StringValue("RowFirst"));
mobility.SetMobilityModel("ns3::RandomWalk2dMobilityModel",
                "Bounds", RectangleValue(Rectangle(-50, 50, -50, 50)));
mobility.Install(wifiStaNodes);

mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(wifiApNode);
// ================= INTERNET =================
InternetStackHelper stack;
stack.Install(csmaNodes);
stack.Install(wifiApNode);
stack.Install(wifiStaNodes);
Ipv4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces = address.Assign(p2pDevices);
address.SetBase("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces = address.Assign(csmaDevices);
address.SetBase("10.1.3.0", "255.255.255.0");
address.Assign(staDevices);
address.Assign(apDevices);
// ================= APPLICATION =================
UdpEchoServerHelper server(9);
ApplicationContainer serverApps = server.Install(csmaNodes.Get(nCsma));
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));
UdpEchoClientHelper client(csmaInterfaces.GetAddress(nCsma), 9);
client.SetAttribute("MaxPackets", UintegerValue(1));
client.SetAttribute("Interval", TimeValue(Seconds(1.0)));
```

```
  client.SetAttribute("PacketSize", UintegerValue(1024));
  ApplicationContainer clientApps =
    client.Install(wifiStaNodes.Get(nWifi - 1));
  clientApps.Start(Seconds(2.0));
  clientApps.Stop(Seconds(10.0));
  Ipv4GlobalRoutingHelper::PopulateRoutingTables();
  // ================= NETANIM =================
  AnimationInterface anim("third-animation.xml");
  anim.SetConstantPosition(p2pNodes.Get(0), 1.0, 2.0);
  anim.SetConstantPosition(p2pNodes.Get(1), 2.0, 2.0);
  // ================= FLOW MONITOR =================
  FlowMonitorHelper flowmon;
  Ptr<FlowMonitor> monitor = flowmon.InstallAll();
  // ================= TRACING =================
  if (tracing)
  {
    pointToPoint.EnablePcapAll("third");
    phy.EnablePcap("third", apDevices.Get(0));
    csma.EnablePcap("third", csmaDevices.Get(0), true);
  }
  // ================= RUN =================
  Simulator::Stop(Seconds(10.0));
  Simulator::Run();
  monitor->SerializeToXmlFile("third-flowmon.xml", true, true);
  Simulator::Destroy();
  return 0;
}
```

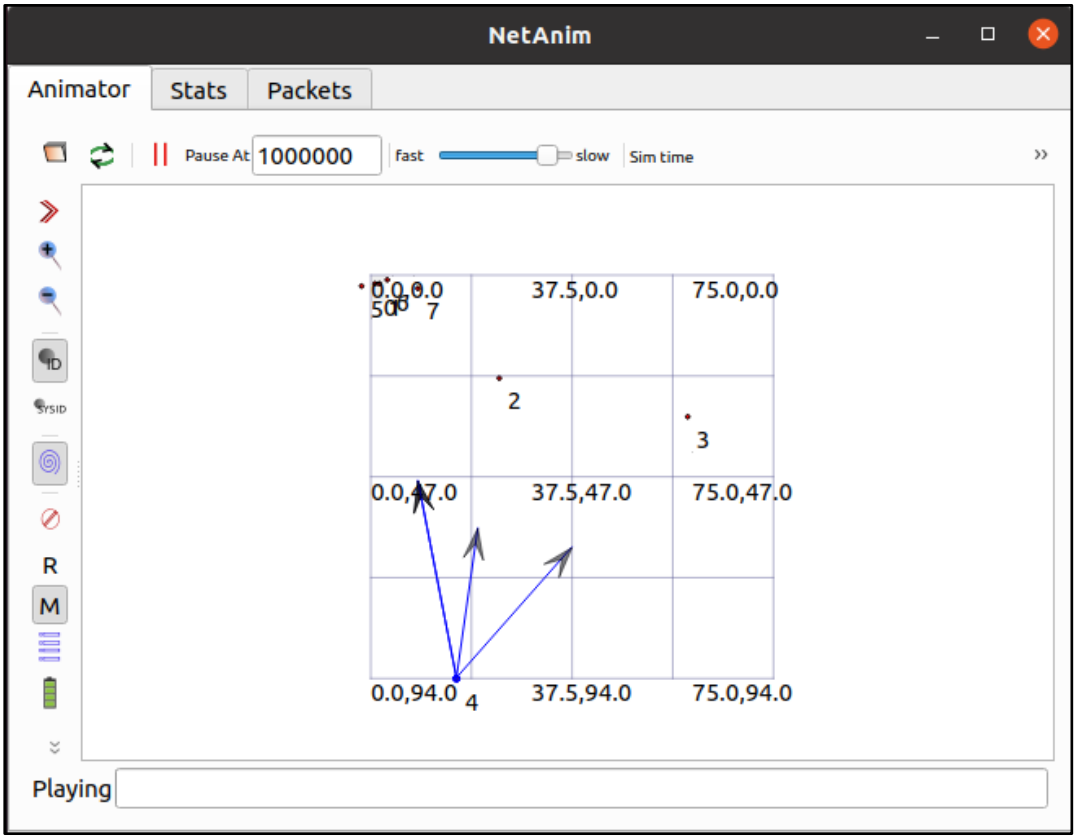
Name:Bhargavikeni-89

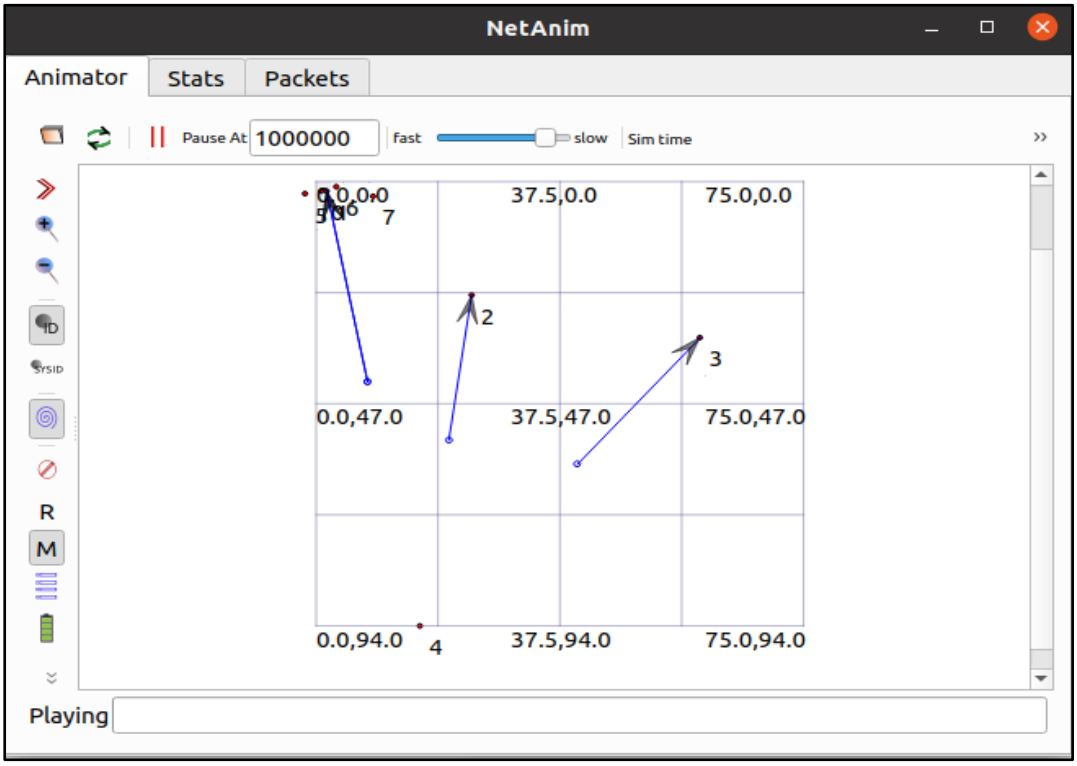
Output:

```
sims@ubuntu:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/hybrid_89.cc --vis
Waf: Entering directory `/home/sims/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/sims/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.306s)
Could not load plugin 'show_last_packets.py': No module named 'kiwi'
Could not load icon applets-screenshooter due to missing gnomedesktop Python module
scanning topology: 8 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
At time +2.02099s server received 1024 bytes from 10.1.3.3 port 49153
At time +2.02099s server sent 1024 bytes to 10.1.3.3 port 49153
At time +2.03771s client received 1024 bytes from 10.1.2.4 port 9
```



Open NetAnim to see the animation:





Practical – 7

Aim: Program to simulate UDP Client Server

Udp_89.cc

```
#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("UdpClientServerExample");
int main(int argc, char *argv[])
{
  // Enable logging for UdpClient and UdpServer
  LogComponentEnable("UdpClient", LOG_LEVEL_INFO);
  LogComponentEnable("UdpServer", LOG_LEVEL_INFO);
  bool useV6 = false;
  Address serverAddress;
  CommandLine cmd;
  cmd.AddValue("useIpv6", "Use Ipv6", useV6);
  cmd.Parse(argc, argv);
  // Create the nodes required by the topology
  NS_LOG_INFO("Create nodes.");
  NodeContainer n;
  n.Create(2);
  InternetStackHelper internet;
  internet.Install(n);
```

```
// Create the channels required by the topology
NS_LOG_INFO("Create channels.");
CsmaHelper csma;
csma.SetChannelAttribute("DataRate", DataRateValue(DataRate(5000000)));
csma.SetChannelAttribute("Delay", TimeValue(MilliSeconds(2)));
csma.SetDeviceAttribute("Mtu", UintegerValue(1400));

NetDeviceContainer d = csma.Install(n);

// Assign 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));
}
// Create Applications
NS_LOG_INFO("Create Applications.");
```

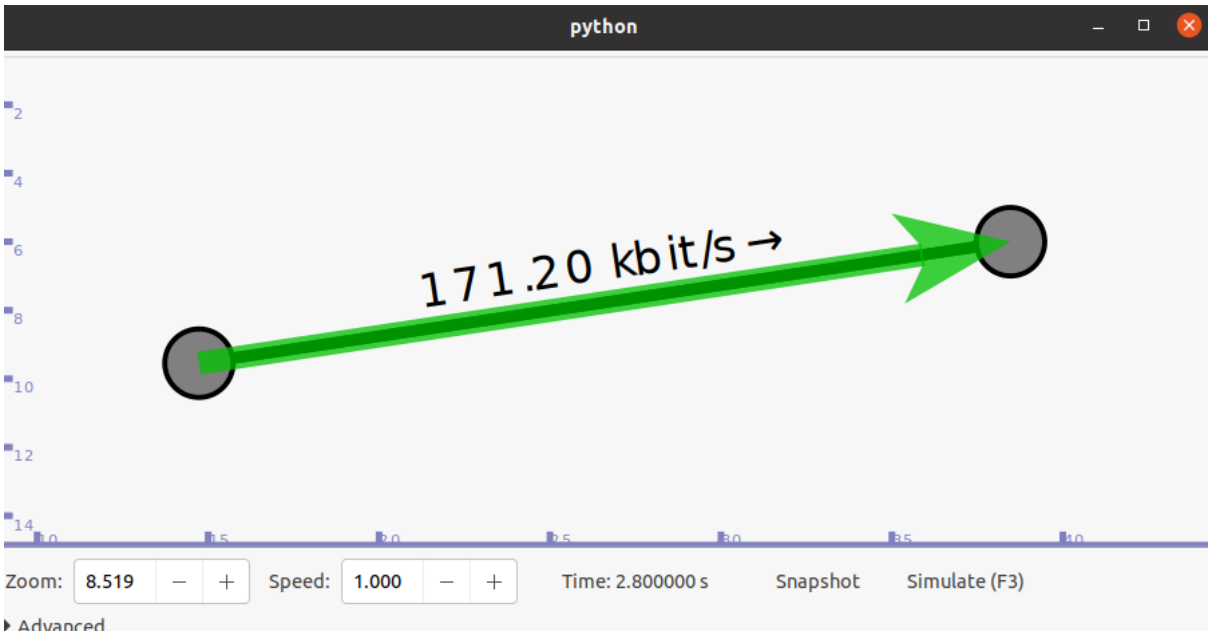
```
  // UDP Server on node 1
  uint16_t port = 4000;
  UdpServerHelper server(port);
  ApplicationContainer apps = server.Install(n.Get(1));
  apps.Start(Seconds(1.0));
  apps.Stop(Seconds(10.0));

  // UDP Client on node 0
  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));
  // Run the simulation
  NS_LOG_INFO("Run Simulation.");
  Simulator::Run();
  Simulator::Destroy();
  NS_LOG_INFO("Done.");
  return 0;
}
```

Name:Bhargavikeni-89

Output:

```
sims@ubuntu:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/udp_89.cc --vis
Waf: Entering directory `/home/sims/ns-allinone-3.32/ns-3.32/build'
[2025/2124] Compiling scratch/udp_89.cc
[2026/2124] Compiling scratch/bus_76.cc
```

```
Waf: Leaving directory `/home/sims/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (16.461s)
```



Practical - 8

Aim: Program to simulate DHCP server and Clients.

Dhcp_89.cc

```
#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"

using namespace ns3;
NS_LOG_COMPONENT_DEFINE("DhcpExample");
int main(int argc, char *argv[])
{
  CommandLine cmd;
  bool verbose = true;
  bool tracing = false;

  cmd.AddValue("verbose", "Turn on the logs", verbose);
  cmd.AddValue("tracing", "Turn on the tracing", tracing);
  cmd.Parse(argc, argv);

  // Enable log components
  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);

// Create nodes
NS_LOG_INFO("Create nodes.");
NodeContainer nodes;
NodeContainer router;

nodes.Create(3);
router.Create(2);

NodeContainer net(nodes, router);

// Create CSMA channel
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);
// Create Point-to-Point connection
NodeContainer p2pNodes;
p2pNodes.Add(net.Get(4)); // router node
p2pNodes.Create(1);
PointToPointHelper pointToPoint;
```

```
pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));
NetDeviceContainer p2pDevices = pointToPoint.Install(p2pNodes);
// Install internet stack
InternetStackHelper tcpip;
tcpip.Install(nodes);
tcpip.Install(router);
tcpip.Install(p2pNodes.Get(1));
// Assign IP addresses for P2P link
Ipv4AddressHelper address;
address.SetBase("172.30.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign(p2pDevices);

// Static 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
);
// DHCP setup
NS_LOG_INFO("Setup DHCP applications.");
```

```
  DhcpHelper dhcpHelper;
  // Fixed IP for router
  Ipv4InterfaceContainer fixedNodes =
    dhcpHelper.InstallFixedAddress(
      devNet.Get(4),
      Ipv4Address("172.30.0.17"),
      Ipv4Mask("/24")
    );
// 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")
    );

  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);
// UDP Echo Server
UdpEchoServerHelper echoServer(9);
ApplicationContainer serverApps =
  echoServer.Install(p2pNodes.Get(1));
serverApps.Start(Seconds(0.0));
serverApps.Stop(stopTime);
// UDP Echo Client
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);
  // Enable tracing
  if (tracing)
  {
    csma.EnablePcapAll("dhcp-csma");
    pointToPoint.EnablePcapAll("dhcp-p2p");
  }

  // Run simulation
  Simulator::Stop(stopTime + Seconds(10.0));
  NS_LOG_INFO("Run Simulation.");
  Simulator::Run();
  Simulator::Destroy();

  NS_LOG_INFO("Done.");
  return 0;
}
```

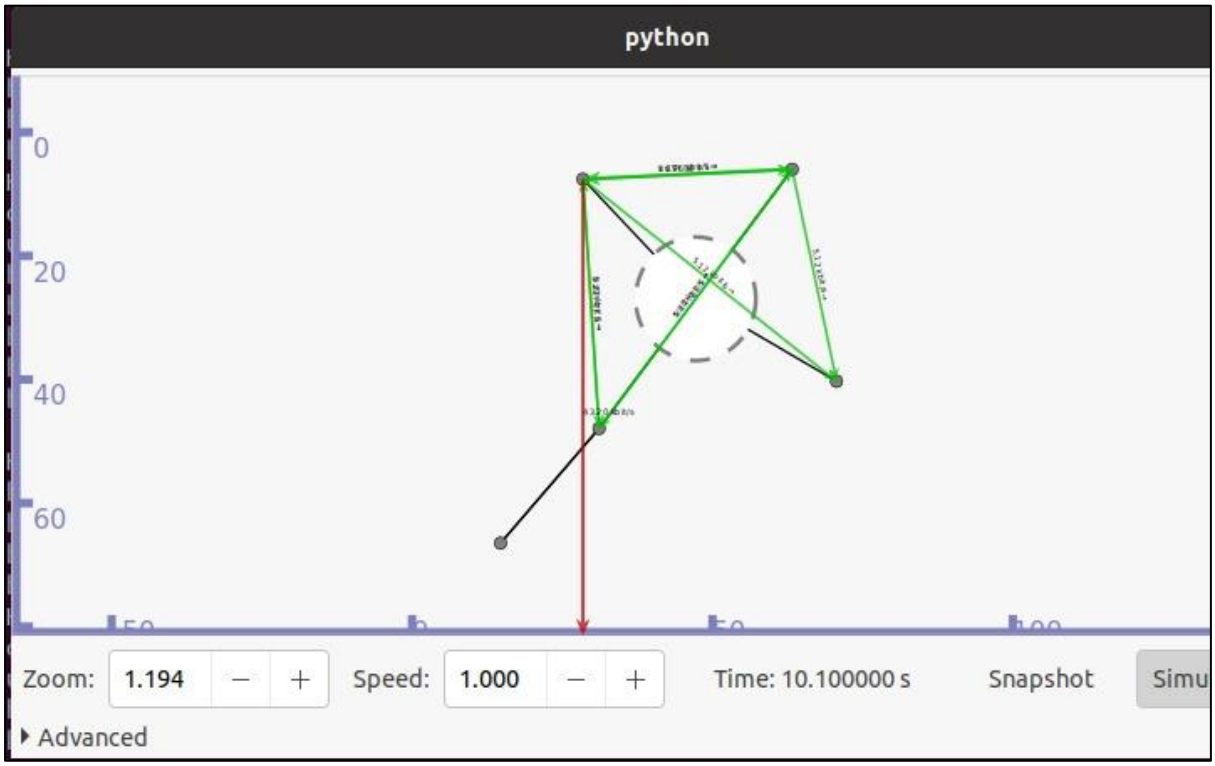
Output:

```
sims@ubntu:~/ns-allinone-3.32/ns-3.32/scratch$ cd ..
sims@ubntu:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/dhcp_89 --vis
Waf: Entering directory `/home/sims/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/sims/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.585s)
[2833/2938] Compiling scratch/point_106.cc
[2834/2938] Compiling scratch/point77.cc
[2835/2938] Compiling scratch/bus.cc
[2836/2938] Compiling scratch/dhcp_89.cc
[2837/2938] Linking build/scratch/point_106
[2838/2938] Linking build/scratch/point77
[2839/2938] Linking build/scratch/bus
[2840/2938] Linking build/scratch/dhcp_89
[2841/2938] Compiling scratch/hybrid.cc
[2842/2938] Compiling scratch/dhcp_89..cc
[2843/2938] Compiling scratch/point.cc
[2844/2938] Linking build/scratch/dhcp_89
[2845/2938] Linking build/scratch/hybrid
[2846/2938] Compiling scratch/mesh106.cc
[2847/2938] Compiling scratch/ftp.cc
[2848/2938] Linking build/scratch/point
[2849/2938] Linking build/scratch/ftp
[2850/2938] Compiling scratch/udp.cc
[2851/2938] Compiling scratch/hybrid106.cc
[2852/2938] Linking build/scratch/mesh106
[2853/2938] Compiling scratch/ftp106.cc
[2854/2938] Linking build/scratch/udp
[2855/2938] Linking build/scratch/hybrid106
[2856/2938] Compiling scratch/Startopo.cc
[2857/2938] Compiling scratch/ftp_106.cc
[2858/2938] Linking build/scratch/Startopo
[2859/2938] Linking build/scratch/ftp_106
```

```
At time +17.0074s server received 1024 bytes from 172.30.0.11 port 49153
At time +17.0074s server sent 1024 bytes to 172.30.0.11 port 49153
At time +17.0148s client received 1024 bytes from 172.30.1.2 port 9
DhcpServer:TimerHandler(0x55582cf0ff80)
At time +18s client sent 1024 bytes to 172.30.1.2 port 9
At time +18.0074s server received 1024 bytes from 172.30.0.11 port 49153
At time +18.0074s server sent 1024 bytes to 172.30.0.11 port 49153
At time +18.0148s client received 1024 bytes from 172.30.1.2 port 9
DhcpServer:TimerHandler(0x55582cf0ff80)
At time +19s client sent 1024 bytes to 172.30.1.2 port 9
At time +19.0074s server received 1024 bytes from 172.30.0.11 port 49153
At time +19.0074s server sent 1024 bytes to 172.30.0.11 port 49153
At time +19.0148s client received 1024 bytes from 172.30.1.2 port 9
DhcpClient:StopApplication(0x55582cf4e2e0)
DhcpClient:StopApplication(0x55582cfd3350)
DhcpClient:StopApplication(0x55582cfd47d0)
DhcpServer:StopApplication(0x55582cf0ff80)
DhcpClient:DoDispose(0x55582cf4e2e0)
DhcpClient:DoDispose(0x55582cfd3350)
DhcpClient:DoDispose(0x55582cfd47d0)
DhcpServer:DoDispose(0x55582cf0ff80)
DhcpClient:~DhcpClient(0x55582cf4e2e0)
DhcpClient:~DhcpClient(0x55582cfd3350)
DhcpClient:~DhcpClient(0x55582cfd47d0)
DhcpServer:~DhcpServer(0x55582cf0ff80)
```

Name:Bhargavikeni-89

Using Python Visualizer



Practical - 9

Aim: Program to simulate FTP using TCP

ftp_89.cc

```
#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"

using namespace ns3;
NS_LOG_COMPONENT_DEFINE("TcpBulkSendExample");
int main(int argc, char *argv[])
{
  bool tracing = false;
  uint32_t maxBytes = 0;

  // Command-line arguments
  CommandLine cmd;
  cmd.AddValue("tracing", "Flag to enable/disable tracing", tracing);
  cmd.AddValue("maxBytes", "Total number of bytes for application to send",
maxBytes);
  cmd.Parse(argc, argv);
  // Create nodes
  NS_LOG_INFO("Create nodes.");
```

```
NodeContainer nodes;
nodes.Create(2);
// Create point-to-point channel
NS_LOG_INFO("Create channels.");
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute("DataRate", StringValue("500Kbps"));
pointToPoint.SetChannelAttribute("Delay", StringValue("5ms"));
NetDeviceContainer devices;
devices = pointToPoint.Install(nodes);
// Install internet stack
InternetStackHelper internet;
internet.Install(nodes);
// Assign IP addresses
NS_LOG_INFO("Assign IP Addresses.");
Ipv4AddressHelper ipv4;
ipv4.SetBase("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer i = ipv4.Assign(devices);
// Create Applications
NS_LOG_INFO("Create Applications.");

// TCP Bulk Send on node 0
uint16_t port = 9; // well-known port
BulkSendHelper source(
   "ns3::TcpSocketFactory",
   InetSocketAddress(i.GetAddress(1), port)
);

// Set total bytes to send (0 = unlimited)
```

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

ApplicationContainer sourceApps = source.Install(nodes.Get(0));
sourceApps.Start(Seconds(0.0));
sourceApps.Stop(Seconds(10.0));

// Packet Sink on node 1
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));

// Enable tracing if required
if (tracing)
{
   AsciiTraceHelper ascii;
   pointToPoint.EnableAsciiAll(ascii.CreateFileStream("tcp-bulk-send.tr"));
   pointToPoint.EnablePcapAll("tcp-bulk-send", false);
}
// Run simulation
NS_LOG_INFO("Run Simulation.");
Simulator::Stop(Seconds(10.0));
Simulator::Run();
Simulator::Destroy();
NS_LOG_INFO("Done.");
```

```
    // Output total bytes received
    Ptr<PacketSink> sink1 = DynamicCast<PacketSink>(sinkApps.Get(0));
    std::cout << "Total Bytes Received: " << sink1->GetTotalRx() << std::endl;

    return 0;
}
```

Output: