# THAKUR COLLEGE OF SCIENCE & COMMERCE

NAAC
Accredited
with Grade "A"
(3rd Cycle)

THAKUR
TRUSTS

ISO
9001 : 2015
Certified

## Degree College

# Computer Journal

## CERTIFICATE

SEMESTER III UID No. 2020858

Class SYBSC CS Roll No. 4334 Year 2021-2022

This is to certify that the work entered in this journal is the work of Mst. / Ms. SHAIKH KAYSAN RAZAUDDIN

who has worked for the year 2021-2022 in the Computer Laboratory.

Teacher In-Charge

Head of Department

Date :

Examiner

# INDEX

# Practical 1

**Aim:** Installation of NS3.

## Theory:

**ns-3** is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free, open-source software, licensed under the GNU GPLv2 license, and maintained by a worldwide community. The goal of the ns-3 project is to develop a free and open source simulation environment suitable for networking research: it should be aligned with the simulation needs of modern networking research and should encouragecommunity contribution, peer review, and validation of the software. ns-3 is maintained by a worldwide team of volunteer maintainers.

## Steps:

Following are the basic steps which must be followed for installing NS3:

1. Install prerequisite packages
2. Download ns3 codes
3. Build ns3
4. Validate ns3

## Prerequisite packages required for Linux:

1. Minimal requirements for Python: gcc g++ python

2. Debugging and GNU Scientific Library (GSL) support: gdbpython-dev

3. valgrind gsl-bin libgsl0-dev libgsl0ldbl Network Simulation Cradle (nsc):flex bison

Reading pcap packet traces: tcpdump

4. Database support for statistics framework: sqlite sqlite3

5. Xml-based version of the config store: libxml2

6. A GTK-based configuration system: libgtk2.0-0

7. Experimental with virtual machines and ns-3: vtun lxc

**Steps:**

1.sudo apt-get update / dnf update 2.sudo apt-

get upgrade / dnf upgrade

3. Once ubuntu/fedora is installed run following command opening the terminal(ctrl+alt+T) window.

4.To install prerequisites dependancy packages- Type the following commandin terminal window.

```
sudo apt-get/ dnf install gcc g++ python python-dev mercurial bzr gdbvalgrind gsl-bin libgsl0-

dev libgsl0ldbl flex bison tcpdump sqlite sqlite3 libsqlite3-dev libxml2libxml2-dev libgtk2.0-0

libgtk2.0-dev uncrustify doxygen graphviz imagemagick texlive texlive- latex-extra texlivegeneric-extra texlive-generic-recommended texinfo dia texlive texlive-latex-extra texlive-extrautils texlive-generic-recommendedtexi2html python-pygraphviz python-kiwi pythonpygoocanvas libgoocanvas-dev python-pygccxml
```

5.After downloading NS3 on the drive, extract all the files in the NS3 folder,which you have created.

6.Then you can find build.py along with other files in NS3 folder.Then to

build the examples in ns-3 run :

```
./build.py --enable-examples –enable-tests
```

If the build is successful then it will give output "Build finished successfully".

7. Now run the following command on the terminal window,to configure withwaf(build tool)

```
./waf -d debug --enable-examples --enable-tests configureTo build
```

with waf(optional)

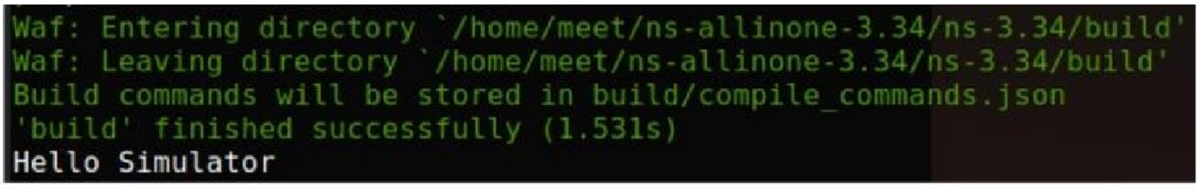
```
./waf
```

8.To test everything allright run the following command on the terminalwindow,

```
./test.py
```

If the tests are ok the installation is done

**Output:**



```
Waf: Entering directory `/home/meet/ns-allinone-3.34/ns-3.34/build'
Waf: Leaving directory `/home/meet/ns-allinone-3.34/ns-3.34/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.531s)
Hello Simulator
```

# Practical 2

**Aim:** Program to connect two nodes using NS3Source.

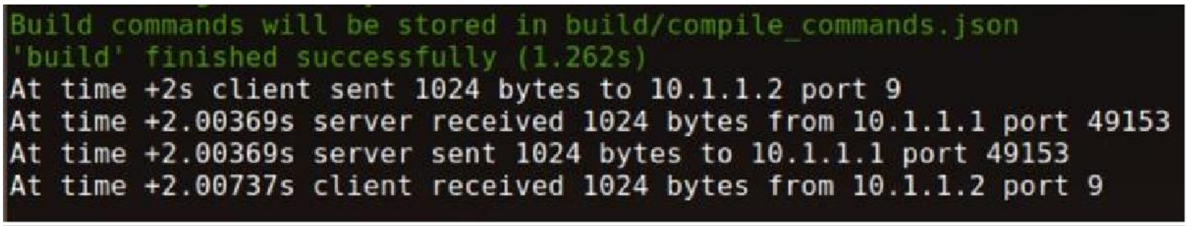
**Code:**

```
#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"

// 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 (1024));

  ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));


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

**Output:**



```
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.262s)
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
```

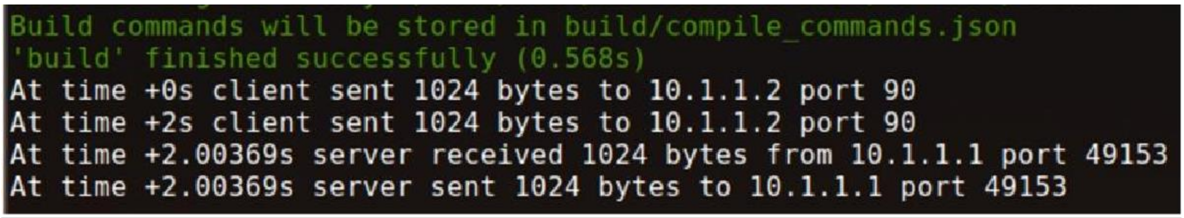
# Practical 3

**Aim:** Program for connecting three nodes considering one node as a centralnode.

**Source Code:**

```
#include "ns3/internet-module.h" #include
"ns3/point-to-point-module.h"#include
"ns3/applications-module.h" using namespace
ns3;
NS_LOG_COMPONENT_DEFINE("FirstScriptExample");int main(int
argc, char *argv[])
{
   Time::SetResolution(Time::NS); LogComponentEnable("UdpEchoClientApplication",
   LOG_LEVEL_INFO); LogComponentEnable("UdpEchoServerApplication",
   LOG_LEVEL_INFO);NodeContainer nodes;
   nodes.Create(3); PointToPointHelper
   pointToPoint;
   pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
   pointToPoint.SetChannelAttribute("Delay", StringValue("2ms")); NetDeviceContainer
   devices, devices1;
   devices = pointToPoint.Install(nodes.Get(0), nodes.Get(1)); devices1 =
   pointToPoint.Install(nodes.Get(2), nodes.Get(1));InternetStackHelper
   stack;
   stack.Install(nodes);
   Ipv4AddressHelper address;
   address.SetBase("10.1.1.0", "255.255.255.0");
   Ipv4InterfaceContainer interfaces = address.Assign(devices); Ipv4InterfaceContainer
   interfaces1 = address.Assign(devices1);
  UdpEchoServerHelper echoServer(90);
   ApplicationContainer serverApps = echoServer.Install(nodes.Get(1));
   serverApps.Start(Seconds(1.0));
   serverApps.Stop(Seconds(10.0));
   UdpEchoClientHelper echoClient(interfaces.GetAddress(1), 90);
   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));
   // UdpEchoClientHelper echoClient(interfaces1.GetAddress(1), 90);
   echoClient.SetAttribute("MaxPackets", UintegerValue(1));
   echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
   echoClient.SetAttribute("PacketSize", UintegerValue(1024));
   ApplicationContainer clientApps1 = echoClient.Install(nodes.Get(2));
   clientApps.Start(Seconds(2.0));
   clientApps.Stop(Seconds(10.0));
   Simulator::Run(); Simulator::Destroy();
   return 0;
}
```

**Output:**



```
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.568s)
At time +0s client sent 1024 bytes to 10.1.1.2 port 90
At time +2s client sent 1024 bytes to 10.1.1.2 port 90
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
```

# Practical 4

**Aim:** Program in NS3 to implement star topology.

**Theory:**

Star topology is a network topology where each individual piece of a network isattached to a central node (often called a hub or switch). The attachment of these network pieces to the central component is visually represented in a form similar to a star. Star topologies also may be implemented with Ethernet/cabled structures, wireless routers and/or other components. In many cases, the central hub is the server, and the additional nodes are clients.

**Source Code:**

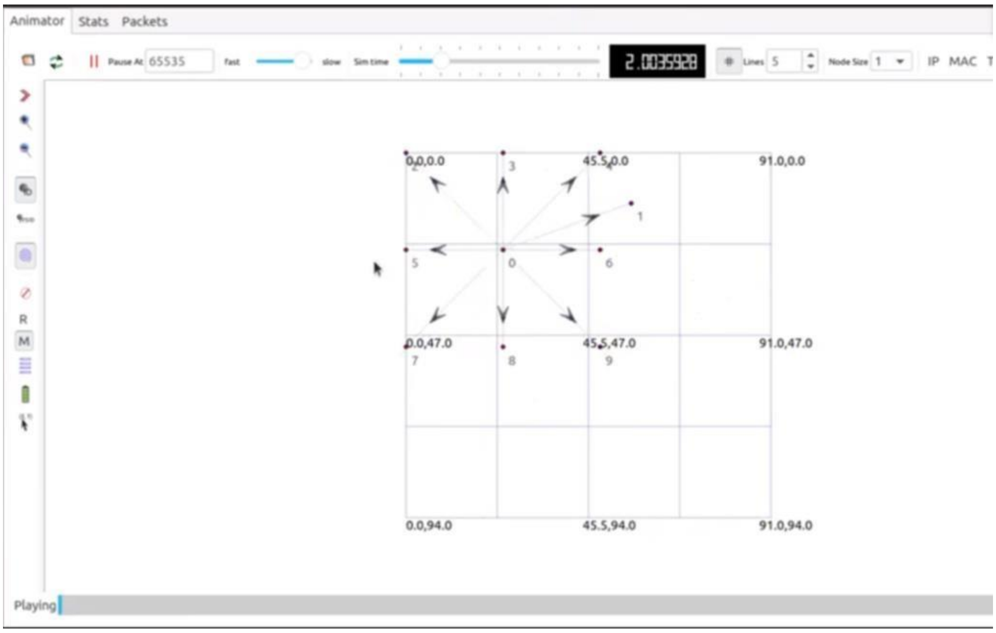
```
#include "ns3/core-module.h" #include

"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"


// 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[])
{

  //
  // 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 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.
  //
  pointToPoint.EnablePcapAll ("star");

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


  return 0;
}
```

**Output:**



# Practical 5

**Aim:**Program in NS3 to implement a bus topology.

**Theory:** A bus network is a local area network (LAN) topology in which each node -- a workstation or other device -- is connected to a main cable or link called a *bus*. All connected stations on the bus can communicate with all otherson the singular network segment.

**Source Code:**

```
#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"


// 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");
```

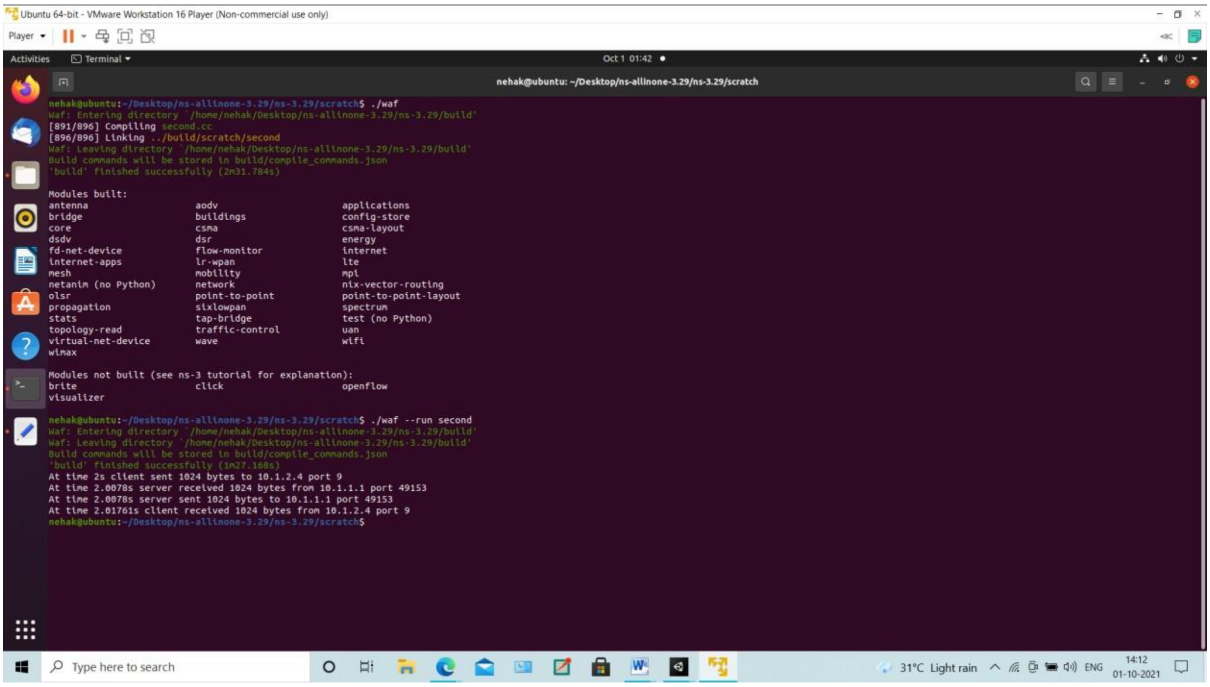
```
int
main (int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 3;

  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);

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

  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces; p2pInterfaces
  = address.Assign (p2pDevices);
```

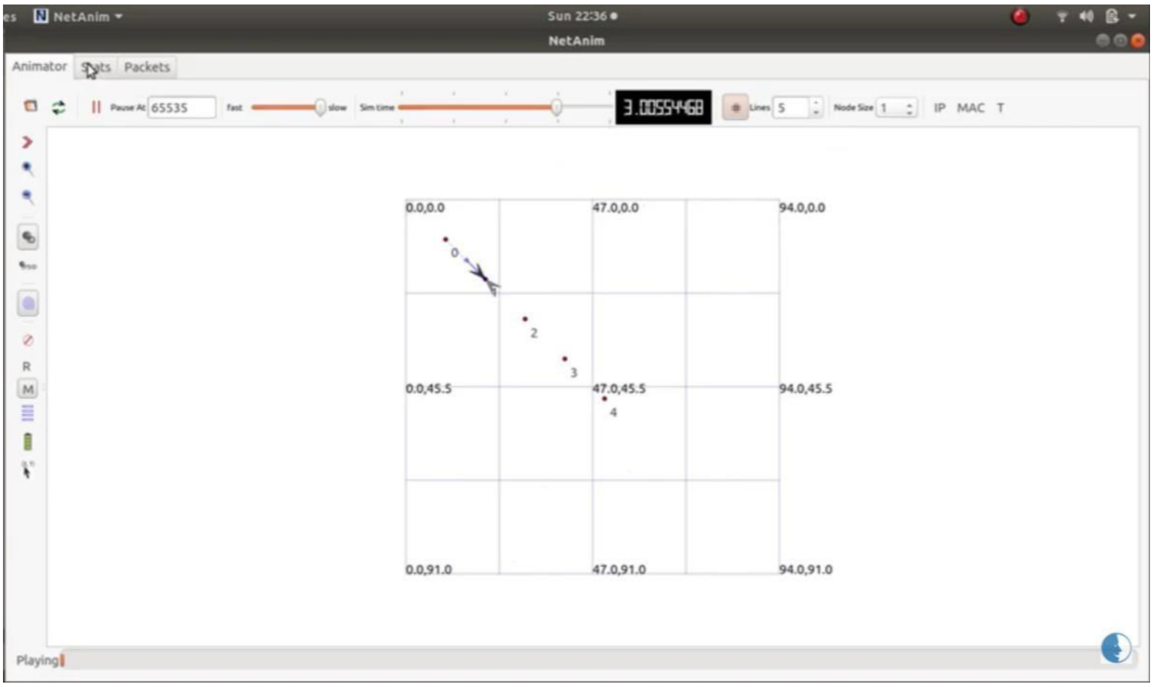
```
address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces; csmaInterfaces =
address.Assign (csmaDevices);

UdpEchoServerHelper echoServer (9);


ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get(nCsma));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));


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

ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));


Ipv4GlobalRoutingHelper::PopulateRoutingTables ();


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

Simulator::Run ();
Simulator::Destroy ();
```

```
 return 0;

}
```

**Output:**


Ubuntu 64-bit - VMware Workstation 16 Player (Non-commercial use only)
Player
Activities
Terminal
Oct 1 01:42
nehak@ubuntu: ~/Desktop/ns-allinone-3.29/ns-3.29/scratch
Type here to search
31°C Light rain
ENG
14:12
01-10-2021

```
nehak@ubuntu:~/Desktop/ns-allinone-3.29/ns-3.29/scratch$ ./waf
Waf: Entering directory `/home/nehak/Desktop/ns-allinone-3.29/ns-3.29/build'
[891/896] Compiling second.cc
[896/896] Linking ../build/scratch/second
Waf: Leaving directory `/home/nehak/Desktop/ns-allinone-3.29/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2m31.784s)

Modules built:
antenna                   aodv                      applications
bridge                    buildings                 config-store
core                      csma                      csma-layout
dsdv                      dsr                       energy
fd-net-device             flow-monitor              internet
internet-apps             lr-wpan                   lte
mesh                      mobility                  mpi
netanim (no Python)       network                   nix-vector-routing
olsr                      point-to-point            point-to-point-layout
propagation               sixlowpan                 spectrum
stats                     tap-bridge                test (no Python)
topology-read             traffic-control           uan
virtual-net-device        wave                      wifi
wimax

Modules not built (see ns-3 tutorial for explanation):
brite                     click                     openflow
visualizer

nehak@ubuntu:~/Desktop/ns-allinone-3.29/ns-3.29/scratch$ ./waf --run second
Waf: Entering directory `/home/nehak/Desktop/ns-allinone-3.29/ns-3.29/build'
Waf: Leaving directory `/home/nehak/Desktop/ns-allinone-3.29/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1m27.168s)
At time 2s client sent 1024 bytes to 10.1.2.4 port 9
At time 2.0078s server received 1024 bytes from 10.1.1.1 port 49153
At time 2.0078s server sent 1024 bytes to 10.1.1.1 port 49153
At time 2.01761s client received 1024 bytes from 10.1.2.4 port 9
nehak@ubuntu:~/Desktop/ns-allinone-3.29/ns-3.29/scratch$
```



# Practical 6

**Aim:** Installation and configuration of NetAnim.

**Theory:** NetAnim is an offline animator based on the Qt toolkit. It currently animates the simulation using an XML trace file collected during simulation. The first version was developed by George F Riley. It is a stand-alone programwhich uses the custom trace files generated by the animation interface to graphically display the simulation. NetAnim is based on the multi-platform Qt4GUI toolkit.

## Source Code:

## Installation:

http://www.nsnam.org/wiki/index.php/NetAnim

1. Install Mercurial:

```
apt−get/dnf install mercurial
```

2. Install QT4 development package:

```
apt−get/dnf install qt4−dev−tools
```

3. You can use Synaptic too, to install both the above packages.

4. Download NetAnim: hg clone http :// code .nsnam. org/netanim

5. Build NetAnim:

```
cd netanim

make clean qmake

NetAnim. pro make
```

## Configuration:

Make the following changes to the code, in order to view the animation onNetAnim.

```
#include " ... "

#include "ns3/netanim−module .h" //1 Include. . .int main

( int argc , char ∗argv [ ] )
```

```
{ std : : string animFile = "somename. xml"; //2 Name of file for animation
. . .
AnimationInterface anim ( animFile ); //3 Animation interfaceSimulator : :
Run ();
Simulator : : Destroy ();
return 0;
}
```

**Run:**

1. Move the waf , waf.bat , wscript and wutils.py les in to the scratchfolder (~/ns-allinone-3.34/ns-3.34/scratch/).

2. Move the example code to the scratch folder and make the changesrequired for NetAnim, as shown above.

3. Now cd to the scratch folder (cd ~/ns-allinone-3.24/ns-3.24/scratch/).

4. Run the code using the command:

```
./ waf −−run <filename>
```

Visualize:

1. cd to the netanim folder (cd ~/netanim/).

2. Run Netanim:

```
./NetAnim
```

3. Include the .xml file generated in the ns-3.24 folder