Practical 1 : Installation of Ubuntu and NS3

Networking with Linux
In this we are going to learn network topology logical working using NS-3 and Wireshark simulator.
This simulator nothing but software model in analogus to hardware working of network.
For. Transmission of Packet over Network node shown in NS-3 GUI window.

To download NS-3 refer below website
https://www.nsnam.org/

https://www.nsnam.org/
Step 1.
On Ubuntu Terminal type command
sudo apt update

Step 2. After fetching latest package install it
command is
sudo apt install

Step 3. After installation upgrade Ubuntu
command is
sudo apt upgrade

Step 4. To Install C++ complier in Ubuntu then follows following command
sudo apt-get install build-essential

Step 5 : To install Python compiler use following command
sudo apt-get install python python-dev

Step 6: To install Mercurial means animation and NS3 supported package use following commands
sudo apt-get install mercurial

Step 7 : To unzip file install supported package
command is
sudo apt-get install bzr

Step 8: to install TCPDUMP for pcap trace file reading support use following command
sudo apt-get install tcpdump

Step 9: to Generate XML supported file to be readble file external simulator API use the following command
sudo apt-get install libxml2 libxml2-dev

Step 10: Unzip the NS3 package
we have command
tar xjf ns-allinone3.32.tar.bz2

Step 11 : To compile all the package in ns-3.32 execute following command

./waf configure --enable-examples –enable-tests

Step 12 : To run our simulator then every classess must be compile . Use the following command

./waf –-run hello-simulator

After executing above commands then output will " Hello Simulator" then assume that all NS3 API are in executable code of Operating System.

$
$
$
$
$
cd
mkdir workspace

cd workspace

wget https://www.nsnam.org/release/ns-allinone-3.32.tar.bz2

tar xjf ns-allinone-3.32.tar.bz2

Notice the use above of the wget utility, which is a command-line tool to fetch objects from the web; if you do not
have this installed, you can use a browser for this step.
Following these steps, if you change into the directory ns-allinone-3.32 , you should see a number of files and
directories

$ cd ns-allinone-3.32
$ ls

bake
constants.py
build.py netanim-3.108
ns-3.32
pybindgen-0.21.0
README
util.py
You are now ready to build the base ns-3 distribution and may skip ahead to the section on building ns-3.

Now go ahead and switch back to the debug build that includes the examples and tests.
$ ./waf clean

to compile all API classes use following commands

$ ./waf configure --build-profile=debug --enable-examples –enable-tests

if you don’t see the “Hello Simulator” output, type
the following:

$ ./waf configure --build-profile=debug --enable-examples –enable-tests

and ten type command

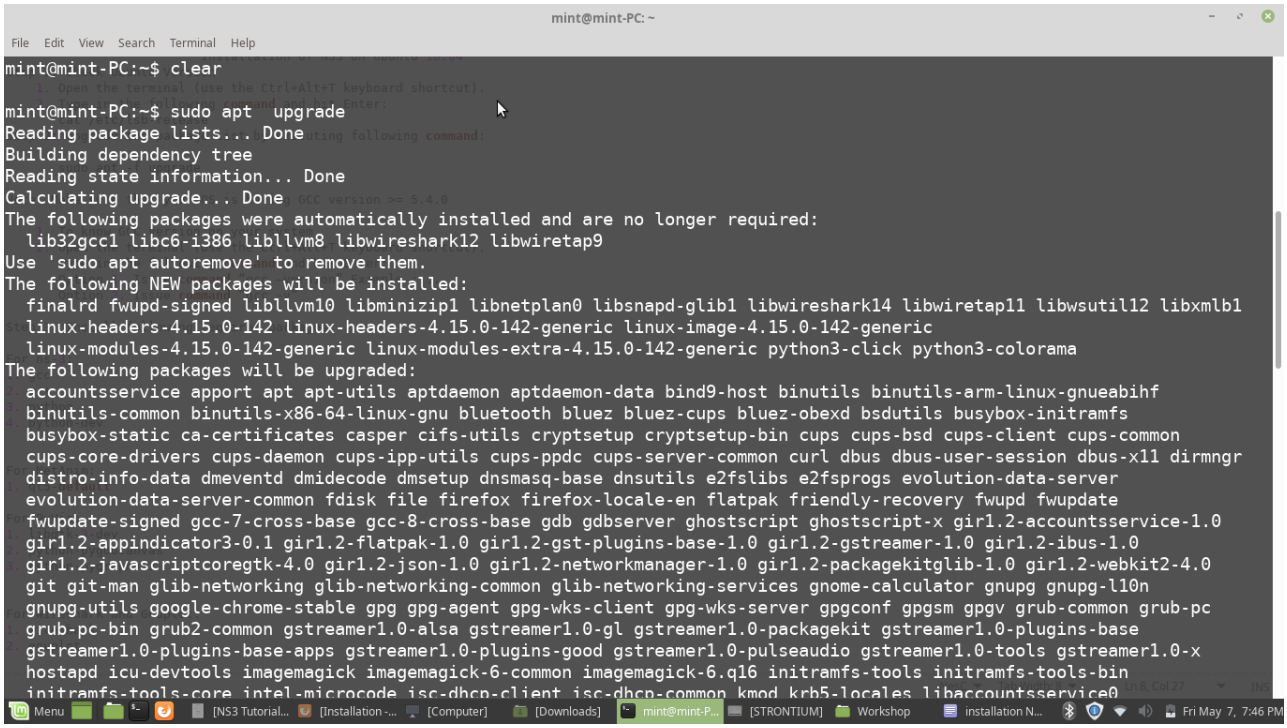
./waf configure –enables-examples
and

./waf --run hello-simulator

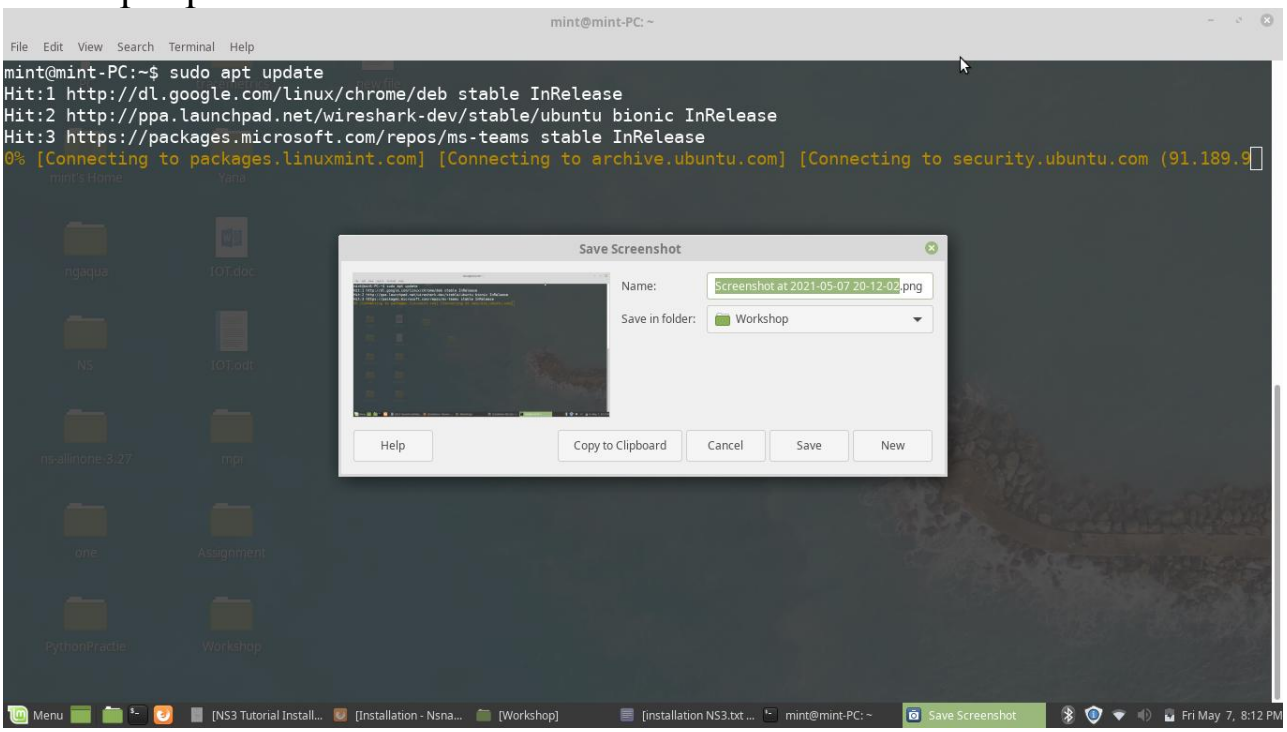
Practical 2: List of Packages for Installing ns-3 in Ubantu/Mint Systems

Perquisite for installing NS3.32

1 sudo apt upgrade



2 Sudo apt update

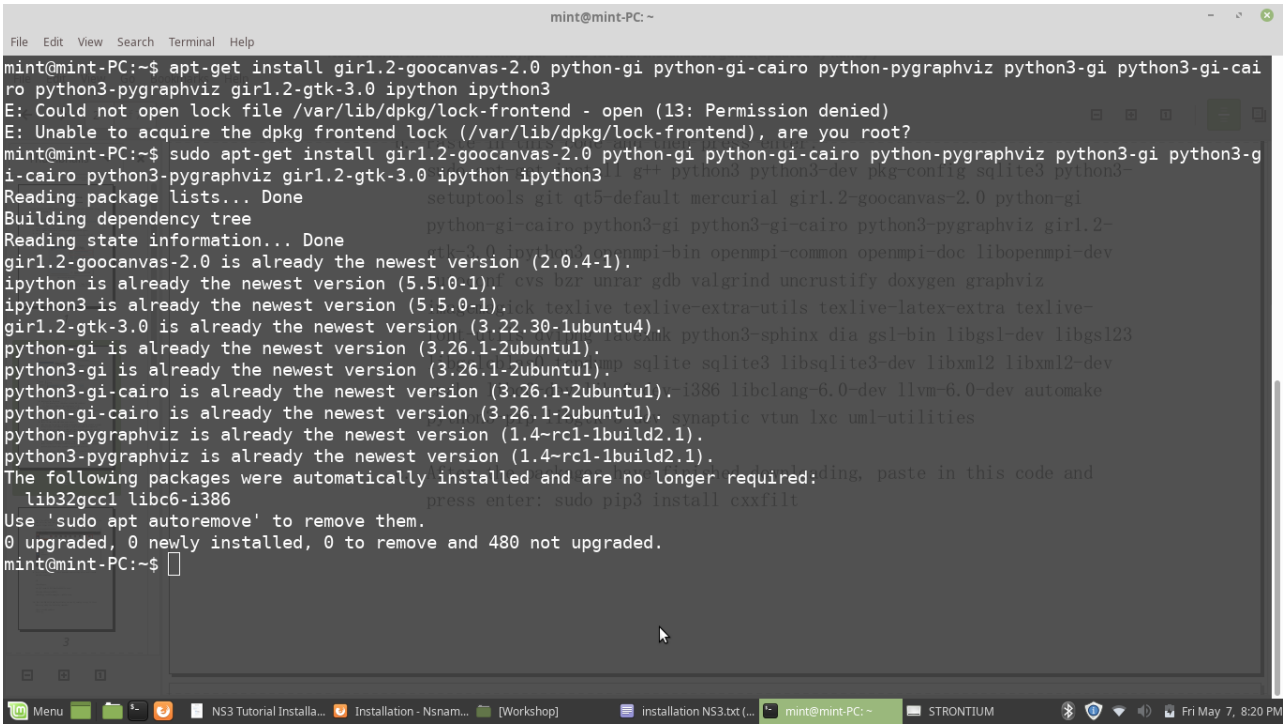


3 Minimal requirements for C++ users

```
apt-get install g++ python3
```

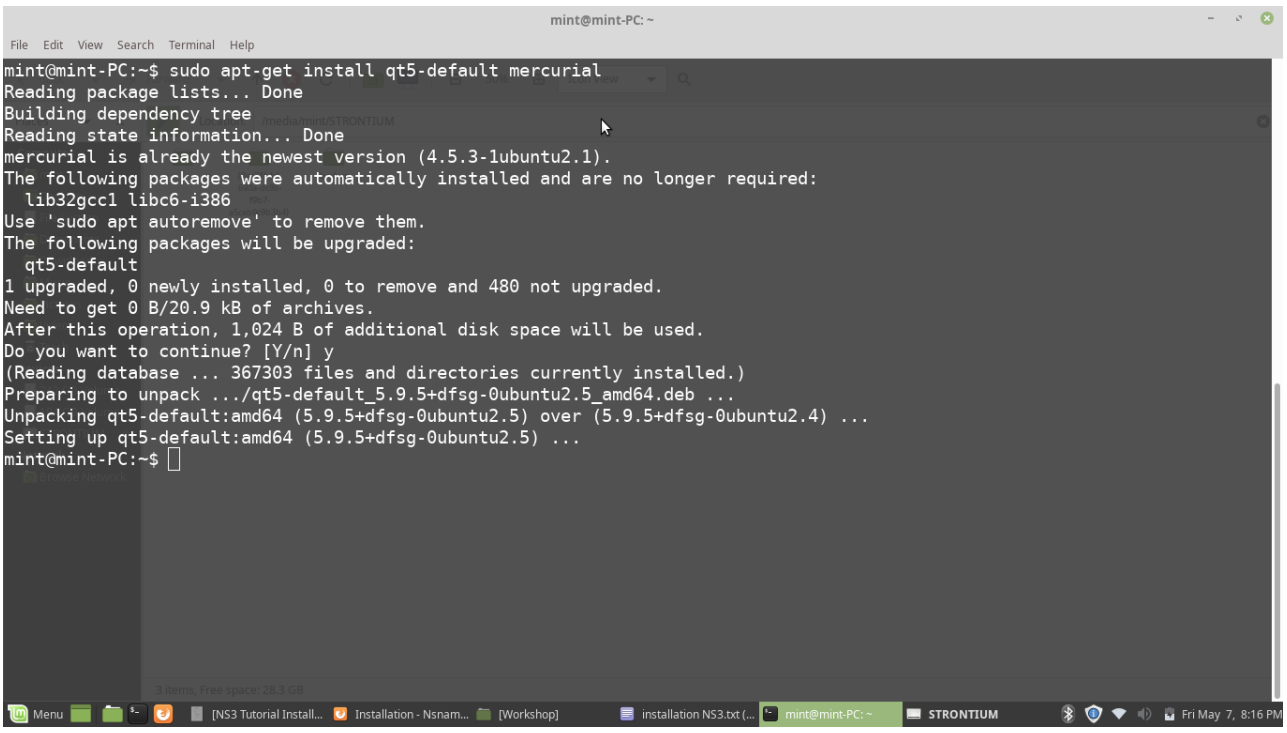
4 Minimal requirements for Python API users

```
apt-get install g++ python3 python3-dev pkg-config sqlite3
```



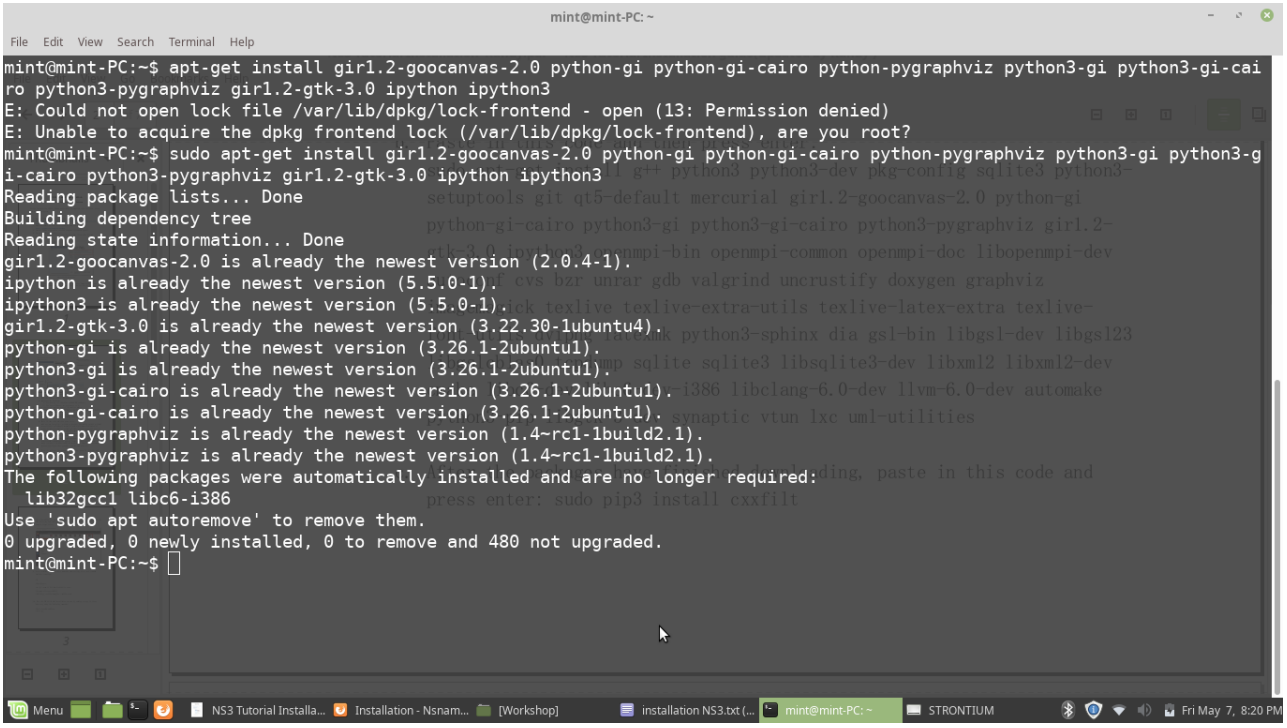
5 **Netanim animator:** `qt5 development tools are needed for Netanim animator;`

```
apt-get install qt5-default mercurial
```



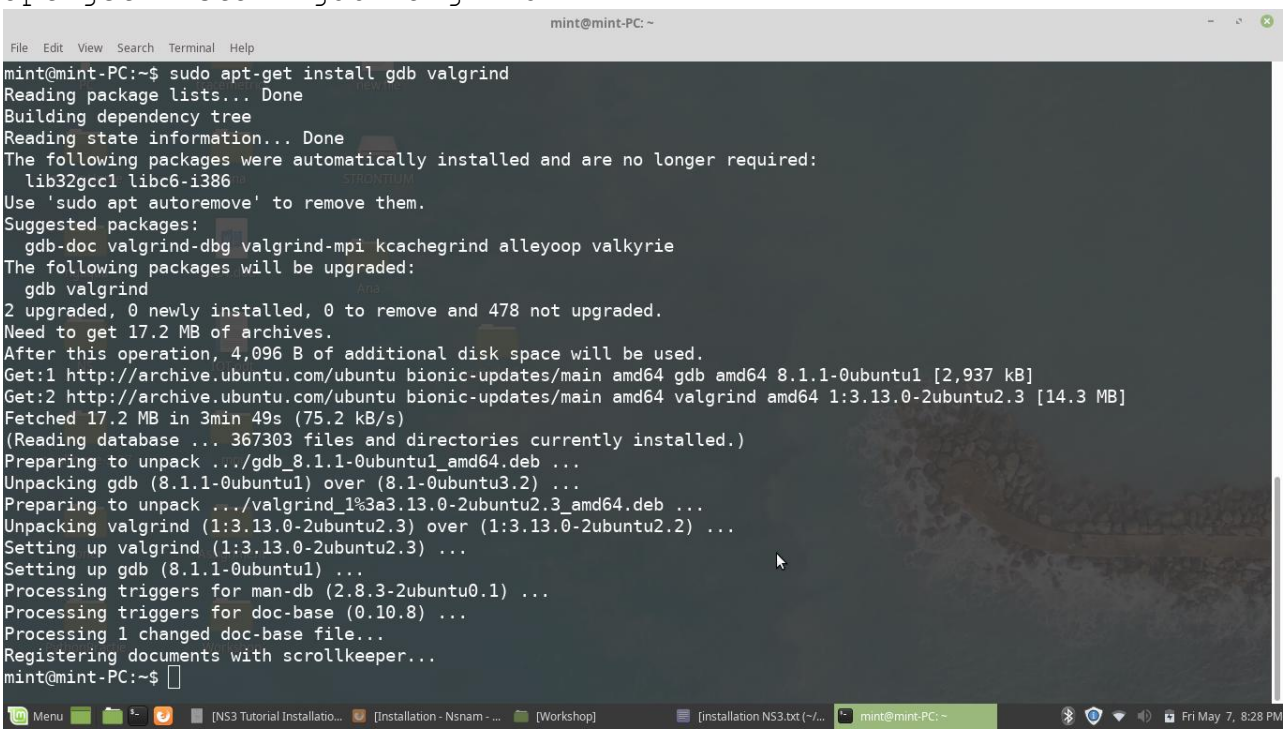
6 ns-3-pyviz visualizer

```
apt-get install gir1.2-goocanvas-2.0 python-gi python-gi-cairo
python-pygraphviz python3-gi python3-gi-cairo python3-pygraphviz
gir1.2-gtk-3.0 ipython ipython3
```



7 Debugging:

8 `apt-get install gdb valgrind`



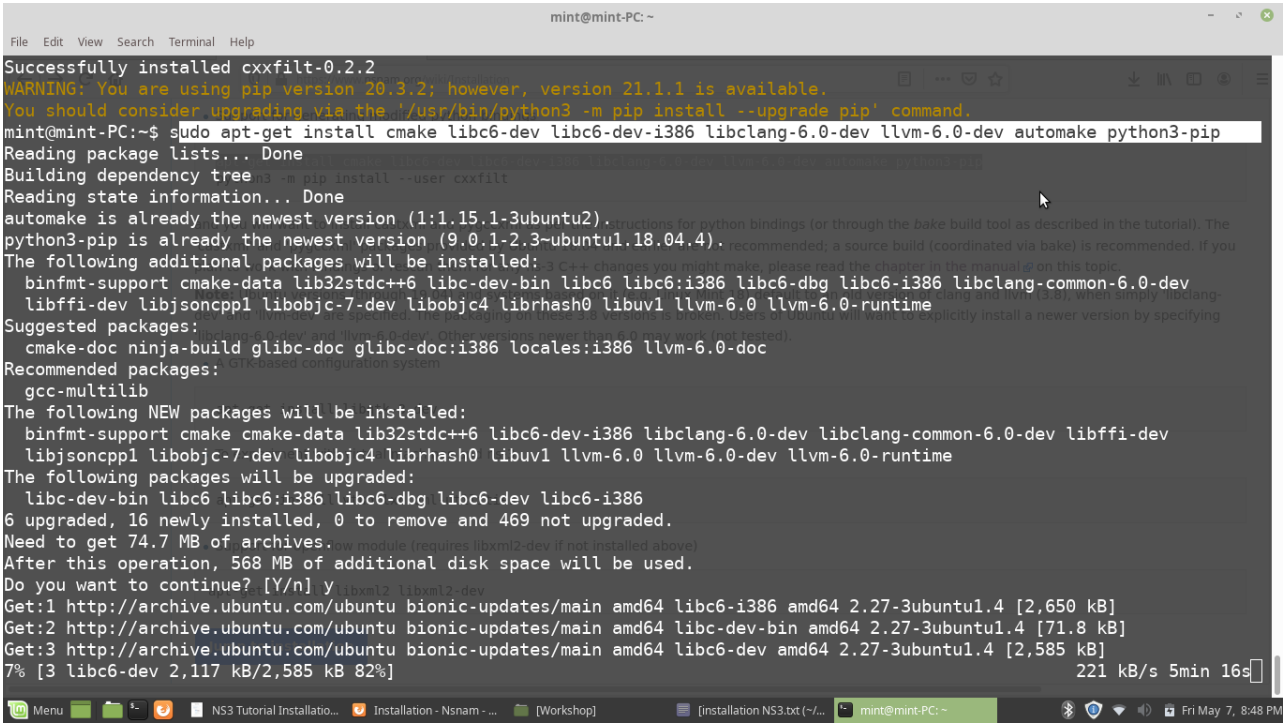
9 Doxygen and related inline documentation:

```
apt-get install doxygen graphviz imagemagick
```

```
apt-get install texlive texlive-extra-utils texlive-latex-extra
texlive-font-utils dvipng latexmk
```

10 **The ns-3 manual and tutorial are written in reStructuredText for Sphinx (doc/tutorial, doc/manual, doc/models), and figures typically in dia (also needs the texlive packages above):**
```
apt-get install python3-sphinx dia
```
11 **To read pcap packet traces**
```
apt-get install tcpdump
```

12 **Support for generating modified python bindings**
```
apt-get install cmake libc6-dev libc6-dev-i386 libclang-6.0-dev
llvm-6.0-dev   automake python3-pip
python3 -m pip install -- user cxxfilt
```



After installing the required packages, create a folder named **workspace** in the home directory and then put the NS3 tar package into the workspace.
Go to terminal and input these commands consecutively after each command finishes executing:
cd
cd workspace
tar xjf <name of NS3 downloaded file name>

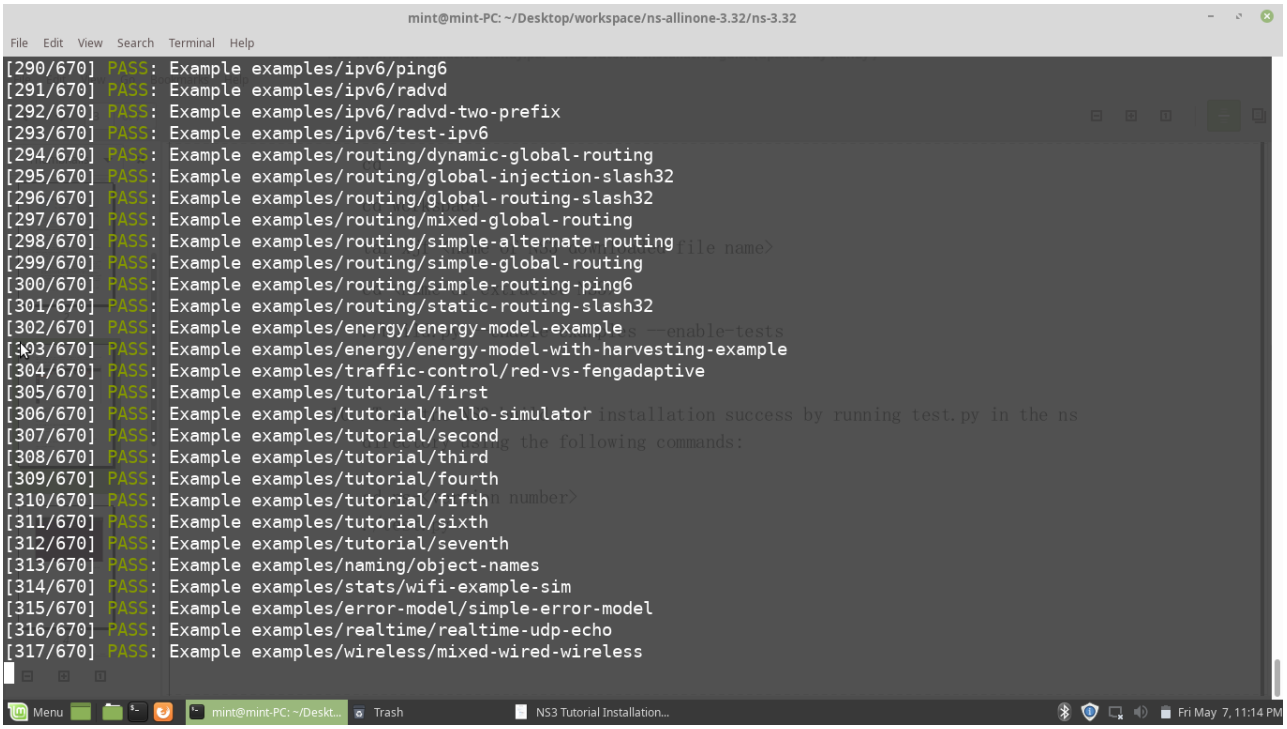
cd <name of extracted NS3>

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

It takes time be patient !!

Test the NS3 build and installation success by running test.py in the ns directory using the following commands:

cd ns-<version number>./test.py



Practical 3: Steps for Installing NetAnim and Wireshark Simulator

Installation of NS3 on Ubuntu 18.04/Linux Mint 19

Step 1: Check Ubuntu Version

1. Open the terminal (use the Ctrl+Alt+T keyboard shortcut).
2. Type in the following command and hit Enter:
   cat /etc/lsb-release

Step 2: Upgrade the package list by executing following command:

sudo apt -f upgrade

Note: Make sure that your OS is having GCC version >= 5.4.0

1. To know GCC version on your system
2. Open the terminal (use the Ctrl+Alt+T keyboard shortcut).
   Type in the following command and hit Enter:
   Option 1. Issue command "gcc –version" Example : ...
   Option 2. Issue command "gcc -v"

Step 3: Install following required packages:

For ns-3:
1. gcc
2. g++
3. python
4. python-dev

For NetAnim:
1. qt5-default

For PyViz:
1. libgtk-3-dev
2. python-pygoocanvas
3. python-pygraphviz

For Wireshark and Gnuplot:

1. wireshark
2. gnuplot

Command to install all these packages together:

sudo apt-get install gcc g++ python python-dev qt4-dev-tools libgtk-3-dev python-pygoocanvas python-pygraphviz wireshark gnuplot

Step 4 : Download ns-allinone-3.27.tar.bz2 and unzip it. For unzip write following command.

tar -xvjf ns-allinone-3.27.tar.bz2

Step 5 : Go to ns-allinone-3.27 folder and install ns3: Follow the bellow command to do so:

cd ns-allinone-3.27

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

This command will install ns-3, NetAnim and PyViz. You will be having a list of built and unbuilt modules after successful installation.

Step 6: Once the installation completes, go to ns-3.27 and give the following command for testing the installation:

cd ns-allinone-3.27/ns-3.27/./test.py -c core

./waf –run hello-simulator

You are done!

Step 7: To install the Wireshark
ns-allinone-3.32$sudo apt-get install wireshark

Step 8: To install GNUplot
ns-allinone-3.32$ sudo apt-get gnuplot

Note :
1. Minimal requirements for C++ users
sudo apt-get install g++ python3

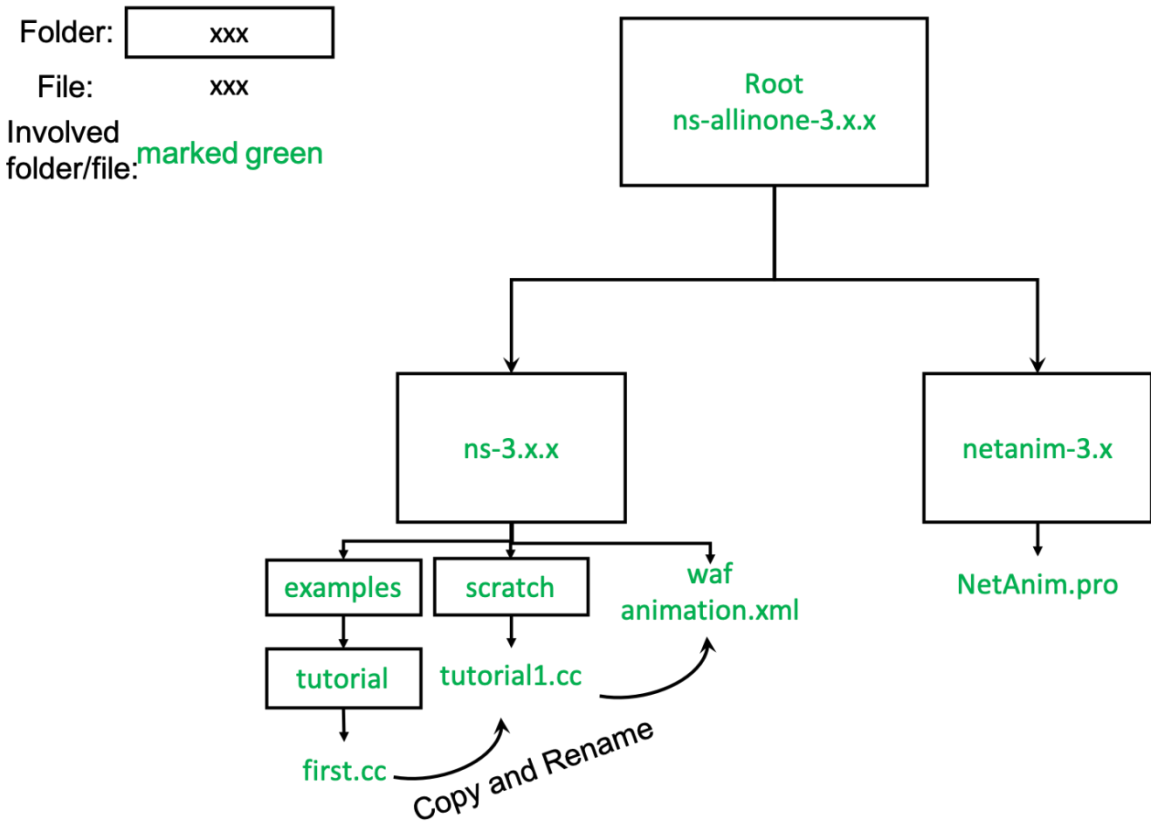
2. Minimal requirements for Python API users
sudo apt-get install g++ python3 python3-dev pkg-config sqlite3

3. NetAnim animator : - qt5 development tools are needed for NetAnim animator
sudo apt-get install qt5-default mercurial

Practical 4: **Configure NetAnim Simulaor**

Prerequests : **NS3 Installation on Ubuntu 1804**

Related folders and files in this document



Configure NetAnim

```
# Go to NS3 All-in-one folder and go further into NetAnim folder
cd ns-allinone-3.30.1/netanim-3.108/

#
sudo apt-get install make -y

# clean the source code, qmake and then make
make clean
qmake NetAnim.pro
make
```

Now, NetAnim is ready to use.

Run Tutorial Examples

Pay attention to the name of the file

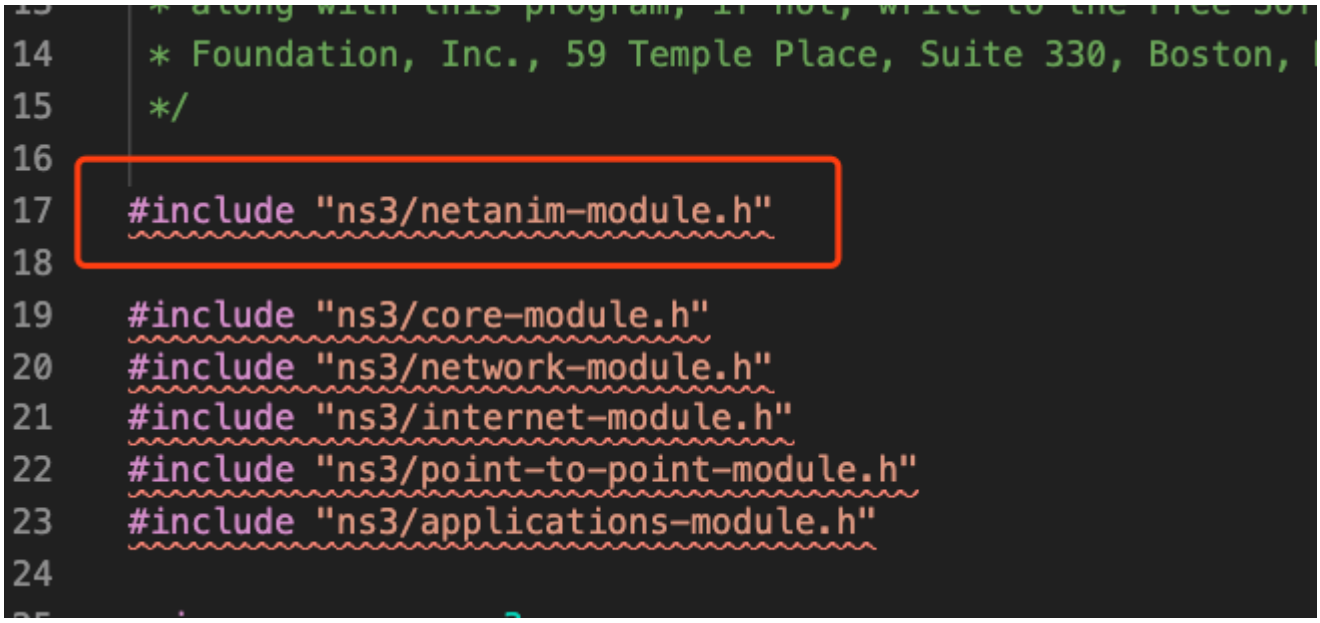
```
# go to ns3-30.1 examples
```

```
cd ..  # now we are at ns-allinone-3.30.1/
cd ns-3.30.1/examples/tutorial

# instead of ruining the example, we copy it to scratch/  Note that, you cannot change the folder unless you have configured the path
# Also, you should not reuse the name "first", because NS3 will find out first.cc or first.py instead of your file.
# here we use tutorial1.cc
cp first.cc ../../scratch/tutorial1.cc
cd ../../scratch/

# use your favourite code editor to edit the first.cc file.
# I use vscode.
```

Add Code for NetAnim

1 Add the header file

#include “ns3/netanim-module.h”



2 Add the following statement before Simulation::Run()

AnimationInterface anim (“animation.xml”);

```
68    ApplicationContainer clientApps = echoClient.Install (nodes.Get (0));
69    clientApps.Start (Seconds (2.0));
70    clientApps.Stop (Seconds (10.0));
71
72    AnimationInterface anim ("animation.xml");
73
74    Simulator::Run ();
75    Simulator::Destroy ();
76    return 0;
77  }
```

3 Set give positions to your nodes.

anim.SetConstantPosition (node, double x, double y);

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

AnimationInterface anim ("animation.xml"); // an xml to be used by netanim
// there are two nodes in this simulation scenario (see "nodes.Create (2);")
anim.SetConstantPosition (nodes.Get(0), 1.0, 2.0); // place node-0 at (1, 2)
anim.SetConstantPosition (nodes.Get(1), 4.0, 5.0); // place node-1 at (4, 5)

Simulator::Run ();
```

2.2.3.6. Build and Run

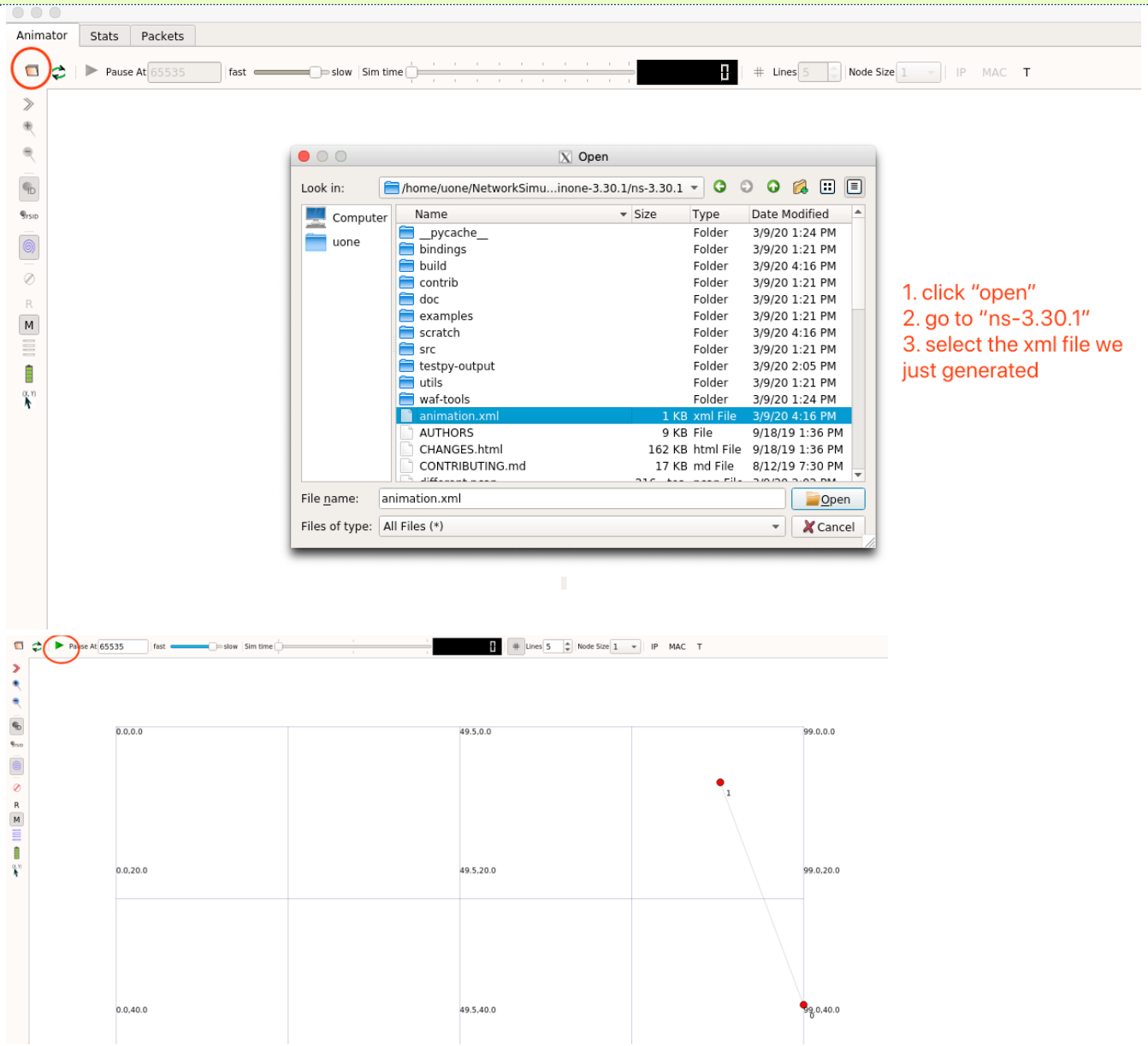
```
# go back to ns-3.30.1
cd ../

# use waf to compile and run the code
./waf --run tutorial1
```

You should see

```
Waf: Entering directory `/home/uone/NetworkSimulator/NS3/ns-allinone-3.30.1/ns-3.30.1/build'
[2748/2822] Compiling scratch/tutorial1.cc
[2750/2822] Compiling scratch/subdir/scratch-simulator-subdir.cc
[2780/2822] Linking build/scratch/subdir/subdir
[2781/2822] Linking build/scratch/tutorial1
Waf: Leaving directory `/home/uone/NetworkSimulator/NS3/ns-allinone-3.30.1/ns-3.30.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (3.101s)
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
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
```

Use NetAnim to Show The Animation

```
# go to netanim-3.108
cd ../netanim-3.108/

# run NetAnim
./NetAnim
```



1. click “open”
2. go to “ns-3.30.1”
3. select the xml file we just generated

Practical 5 : Conceptual Explaination of NS3 code

Conceptual Overview

Node

In Internet jargon, a computing device that connects to a network is called a host or sometimes an end system. Because |ns3| is a network simulator, not specifically an Internet simulator, we intentionally do not use the term host since it is closely associated with the Internet and its protocols. Instead, we use a more generic term also used by other simulators that originates in Graph Theory --- the node.

In |ns3| the basic computing device abstraction is called the node. This abstraction is represented in C++ by the class Node.

The Node class provides methods for managing the representations of computing devices in simulations.
You should think of a Node as a computer to which you will add functionality. One adds things like applications, protocol stacks and peripheral cards with their associated drivers to enable the computer to do useful work. We use the same basic model in |ns3|.

Application

Typically, computer software is divided into two broad classes. System Software organizes various computer resources such as memory, processor cycles, disk, network, etc., according to some computing model. System software usually does not 1use those resources to complete tasks that directly benefit a user. A user would typically run an application that acquires and uses the resources controlled by the system software to accomplish some goal.
Often, the line of separation between system and application software is made at the privilege level change that happens in operating system traps. In |ns3| there is no real concept of operating system and especially no concept of privilege levels or system calls. We do, however, have the idea of an application. Just as software applications run on computers to perform tasks in the "real world," |ns3| applications run on |ns3|
Nodes to drive simulations in the simulated world.

In |ns3| the basic abstraction for a user program that generates

some activity to be simulated is the application.

This abstraction is represented in C++ by the class Application.
The Application class provides methods for managing the
representations of our version of user-level applications in
simulations.

Developers are expected to specialize the Application class in the object-oriented programming sense to create new applications.

In this tutorial, we will use specializations of class Application called UdpEchoClientApplication and UdpEchoServerApplication.

As you might expect, these applications compose a client/server application set used to generate and echo simulated network packets

Channel
In the real world, one can connect a computer to a network.
Often the media over which data flows in these networks are
called channels. When you connect your Ethernet cable to the
plug in the wall, you are connecting your computer to an
Ethernet communication channel. In the simulated world
of |ns3|, one connects a Node to an object representing a
communication channel.

Here the basic communication subnetwork abstraction is called the channel and is represented in C++ by the class Channel.

The Channel class provides methods for managing
communication subnetwork objects and connecting nodes to
them. Channels may also be specialized by developers in the
object oriented programming sense.

A Channel specializationmay model something as simple as a wire. The
specialized Channel can also model things as complicated as a
large Ethernet switch, or three-dimensional space full of
obstructions in the case of wireless networks.

We will use specialized versions of the Channel called CsmaChannel, PointToPointChannel and WifiChannel.

The CsmaChannel, for example,
models a version of a communication subnetwork that
implements a carrier sense multiple access communication
medium. This gives us Ethernet-like functionality.

Net Device

It used to be the case that if you wanted to connect a computer to a network, you had to buy a specific kind of network cable and a hardware device called (in PC terminology) a peripheral card that needed to be installed in your computer. If the peripheral card implemented some networking function, they were called Network Interface Cards, or NICs.

Today mostcomputers come with the network interface hardware built in and users don't see these building blocks.

A NIC will not work without a software driver to control the hardware. In Unix (or Linux), a piece of peripheral hardware is classified as a device. Devices are controlled using device drivers, and network devices (NICs) are controlled using network device drivers collectively known as net devices. In Unix and Linux you refer to these net devices by names such as eth0.

In |ns3| the net device abstraction covers both the software driver and the simulated hardware. A net device is "installed" in a Node in order to enable the Node to communicate with other Nodes in the simulation via Channels.

Just as in a real computer, a Node may be connected to more than one Channel via multiple NetDevices.

The net device abstraction is represented in C++ by the class NetDevice.

The NetDevice class provides methods for managing connections to Node and Channel objects; and may be specialized by developers in the object-oriented programming sense. We will use the several specialized versions of the NetDevice called CsmaNetDevice, PointToPointNetDevice, and WifiNetDevice.

Just as an Ethernet NIC is designed to work with an Ethernet network, the CsmaNetDevice is designed to work with a CsmaChannel; the PointToPointNetDevice is designed to work with a PointToPointChannel and a WifiNetNevice is designed to work with a WifiChannel.

Topology Helpers

In a real network, you will find host computers with added (or

built-in) NICs.

In |ns3| we would say that you will find Nodes with attached NetDevices. In a large simulated network you will need to arrange many connections between Nodes, NetDevices and Channels.

Since connecting NetDevices to Nodes, NetDevices to Channels, assigning IP addresses, etc., are such common tasks in |ns3|, we provide what we call topology helpers to make this as easy as possible.

For example, it may take many distinct |ns3| core operations to create a NetDevice, add a MAC address, install that net device on a Node, configure the node's protocol stack, and then connect the NetDevice to a Channel. Even more operations would be required to connect multiple devices onto multipoint channels and then to connect individual networks together into internetworks.

We provide topology helper objects that combine those many distinct operations into an easy to use model for your convenience.

A First ns-3 Script

4If you downloaded the system as was suggested above, you will have a release of |ns3| in a directory called repos under your home directory. Change into that release directory, and you should find a directory structure something like the following:

```
AUTHORS examples scratch utils waf.bat* bindings LICENSE
src utils.py waf-tools build ns3
test.py*
utils.pyc wscript
CHANGES.html README
testpy-output VERSION
wutils.py
doc
RELEASE_NOTES testpy.supp waf*
wutils.pyc
```

Change into the examples/tutorial directory. You should see a file named first.cc located there. This is a script that will create a simple point-to-point link between two nodes and echo a single packet between the nodes. Let's take a look at that script line by line, so go ahead and open first.cc in your favorite editor.

Boilerplate

The first line in the file is an emacs mode line. This tells emacs about the formatting conventions (coding style) we use in our source code.

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*-
*/
```

This is always a somewhat controversial subject, so we might as well get it out of the way immediately. The |ns3| project, like most large projects, has adopted a coding style to which all contributed code must adhere. If you want to contribute your code to the project, you will eventually have to conform to the |ns3| coding standard as described in the file doc/codingstd.txt or shown on the project web page here.

We recommend that you, well, just get used to the look and feel of |ns3| code and adopt this standard whenever you are working with our code. All of the development team and contributors have done so with various amounts of grumbling. 5The emacs mode line above makes it easier to get the formatting correct if you use the emacs editor.

The |ns3| simulator is licensed using the GNU General Public License. You will see the appropriate GNU legalese at the head of every file in the |ns3| distribution. Often you will see a copyright notice for one of the institutions involved in the |ns3| project above the GPL text and an author listed below.

```
/*
* 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
*/

Module Includes
The code proper starts with a number of include statements.

```
#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"
```

To help our high-level script users deal with the large number
of include files present in the system, we group includes
according to relatively large modules. We provide a single
include file that will recursively load all of the include files used
in each module. Rather than having to look up exactly what
header you need, and possibly have to get a number of
dependencies right, we give you the ability to load a group of
files at a large granularity. This is not the most efficient
approach but it certainly makes writing scripts much easier.
Each of the |ns3| include files is placed in a directory
called ns3 (under the build directory) during the build process
to help avoid include file name collisions.

The ns3/core-module.h file corresponds to the ns-3 module you will find in
the directory src/core in your downloaded release distribution.
If you list this directory you will find a large number of header
files.

When you do a build, Waf will place public header files in
an ns3 directory under the appropriate build/debug or build/optimized directory
depending on your configuration. Waf will also automatically
generate a module include file to load all of the public header
files.

Since you are, of course, following this tutorial religiously, you
will already have done a

```
$ ./waf -d debug --enable-examples --enable-tests configure
```

in order to configure the project to perform debug builds that
include examples and tests. You will also have done a

```
$ ./waf
```

to build the project.

So now if you look in the directory ../../build/debug/ns3

you will find the four module include files shown above. You can take a look at the contents of these files and find that they do include all of the public include files in their respective modules.

Ns3 Namespace
The next line in the first.cc script is a namespace declaration.

using namespace ns3;
The |ns3| project is implemented in a C++ namespace
called ns3.
This groups all |ns3|-related declarations in a scope
outside the global namespace, which we hope will help with
integration with other code.
The C++ using statement introduces the |ns3| namespace into the current (global) declarative region.
This is a fancy way of saying that after this
declaration, you will not have to type ns3:: scope resolution
operator before all of the |ns3| code in order to use it.

If you are unfamiliar with namespaces, please consult almost any C++
tutorial and compare the ns3 namespace and usage here with
instances of the std namespace and the using namespace
std; statements you will often find in discussions of cout and streams.

Logging

The next line of the script is the following,
NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

We will use this statement as a convenient place to talk about
our Doxygen documentation system.
If you look at the project web site, ns-3 project, you will find a link to "Documentation" in the navigation bar. If you select this link, you will be taken to our documentation page.

There is a link to "Latest Release" that will take you to the documentation for the latest stable release of |ns3|.
If you select the "API Documentation" link, you will be taken to the |ns3| API documentation page.

Along the left side, you will find a graphical representation of
the structure of the documentation.
A good place to start is the NS-3 Modules "book" in the |ns3| navigation tree.

If you expand Modules you will see a list of |ns3| module
documentation. The concept of module here ties directly into
the module include files discussed above.

The |ns3| logging subsystem is discussed in the :ref:`UsingLogging` section, so
we'll get to it later in this tutorial, but you can find out about the
8above statement by looking at the Core module, then expanding
the Debugging tools book, and then selecting the Logging page.

Click on Logging.
You should now be looking at the Doxygen documentation for
the Logging module.
In the list of Macros's at the top of the page you will see the entry
for NS_LOG_COMPONENT_DEFINE.

Before jumping in, it would probably be good to look for the "Detailed Description"
of the logging module to get a feel for the overall operation. You
can either scroll down or select the "More..." link under the
collaboration diagram to do this.

Once you have a general idea of what is going on, go ahead and
take a look at the specific NS_LOG_COMPONENT_DEFINE documentation.

It summarize, this line declares a logging component
called FirstScriptExample that allows you to enable and disable
console message logging by reference to the name.

Main Function
The next lines of the script you will find are,

```
int
main (int argc, char *argv[])
{
```

This is just the declaration of the main function of your
program (script). Just as in any C++ program, you need to
define a main function that will be the first function run. There
is nothing at all special here. Your |ns3| script is just a C++
program.

The next line sets the time resolution to one nanosecond, which happens to be the default value:

```
Time::SetResolution (Time::NS);
```

The resolution is the smallest time value that can be represented (as well as the smallest representable difference 9between two time values).

You can change the resolution exactly once. The mechanism enabling this flexibility is somewhat memory hungry, so once the resolution has been set explicitly we release the memory, preventing further updates.

(If you don't set the resolution explicitly, it will default to one nanosecond, and the memory will be released when the simulation starts.)

The next two lines of the script are used to enable two logging components that are built into the Echo Client and Echo Server applications:

```
LogComponentEnable("UdpEchoClientApplication",
LOG_LEVEL_INFO);

LogComponentEnable("UdpEchoServerApplication",
LOG_LEVEL_INFO);
```

If you have read over the Logging component documentation you will have seen that there are a number of levels of logging verbosity/detail that you can enable on each component.

These two lines of code enable debug logging at the INFO level for echo clients and servers. This will result in the application printing out messages as packets are sent and received during the simulation.

Now we will get directly to the business of creating a topology and running a simulation. We use the topology helper objects to make this job as easy as possible.

Topology Helpers

NodeContainer

The next two lines of code in our script will actually create the |ns3| Node objects that will represent the computers in the

simulation.

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

Let's find the documentation for the NodeContainer class before we continue. Another way to get into the documentation for a given class is via the Classes tab in the Doxygen pages.

If you still have the Doxygen handy, just scroll up to the top of the page and select the Classes tab. You should see a new set of tabs appear, one of which is Class List.

Under that tab you will see a list of all of the |ns3| classes. Scroll down, looking for ns3::NodeContainer.

When you find the class, go ahead and select it to go to the documentation for the class.

You may recall that one of our key abstractions is the Node.

This represents a computer to which we are going to add things like protocol stacks, applications and peripheral cards.

The NodeContainer topology helper provides a convenient way to create, manage and access any Node objects that we create in order to run a simulation.

The first line above just declares a **NodeContainer** which we call nodes. The second line calls the Create method on the nodes object and asks the container to create two nodes.

As described in the Doxygen, the container calls down into the |ns3| system proper to create two Node objects and stores pointers to those objects internally.

The nodes as they stand in the script do nothing. The next step in constructing a topology is to connect our nodes together into a network.

The simplest form of network we support is a single point-to-point link between two nodes. We'll construct one of those links here.

PointToPointHelper

We are constructing a point to point link, and, in a pattern which will become quite familiar to you, we use a topology helper object to do the low-level work required to put the link together.

Recall that two of our key abstractions are the NetDevice and the Channel.

In the real world, these terms correspond roughly to peripheral cards and network cables.

Typically these two things are intimately tied together and one cannot expect to interchange, for example, Ethernet devices and wireless channels.

Our Topology Helpers follow this intimate coupling and therefore you will use a single PointToPointHelper to configure and connect |ns3| PointToPointNetDevice and PointToPointChannel objects in this script.
The next three lines in the script are,

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

The first line, PointToPointHelper pointToPoint;

instantiates a PointToPointHelper object on the stack. From a high-level perspective the next line,

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

tells the PointToPointHelper object to use the value "5Mbps" (five megabits per second) as the "DataRate" when it creates a PointToPointNetDevice object.

From a more detailed perspective, the string "DataRate" corresponds to what we call an Attribute of the PointToPointNetDevice.

If you look at the Doxygen for class ns3::PointToPointNetDevice and find the documentation for the GetTypeId method, you will find a list of Attributes defined for the device. Among these is the "DataRate" Attribute.

Most user-visible |ns3| objects have similar lists of Attributes. We use this mechanism to easily configure simulations without recompiling as you will see in a following section.

Similar to the "DataRate" on the PointToPointNetDevice you will find a "Delay" Attribute associated with the PointToPointChannel.

The final line,
pointToPoint.SetChannelAttribute ("Delay", StringValue("2ms"));
tells the PointToPointHelper to use the value "2ms" (two milliseconds) as the value of the propagation delay of every point to point channel it subsequently creates.

NetDeviceContainer
At this point in the script, we have a NodeContainer that contains two nodes.

We have a PointToPointHelper that is primed and ready to make PointToPointNetDevices and wire PointToPointChannel objects between them.

Just as we used the NodeContainer topology helper object to create the Nodes for our simulation, we will ask the PointToPointHelper to do the work involved in creating, configuring and installing our devices for us.

We will need to have a list of all of the NetDevice objects that are created, so we use a NetDeviceContainer to hold them just as we used a NodeContainer to hold the nodes we created.

The following two lines of code,

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

will finish configuring the devices and channel. The first line declares the device container mentioned above and the second does the heavy lifting.

The Install method of the PointToPointHelper takes a NodeContainer as a parameter.

Internally, a NetDeviceContainer is created.

For each node in the NodeContainer (there must be exactly two for a point-to-point link) a PointToPointNetDevice is created and saved in the device container.
A PointToPointChannel is created and the two PointToPointNetDevices are attached. When objects are created by the PointToPointHelper, the Attributes previously set in the helper are used to initialize the corresponding Attributes in the created objects.

After executing the pointToPoint.Install (nodes) call we will
have two nodes, each with an installed point-to-point net device
and a single point-to-point channel between them.

Both devices will be configured to transmit data at five megabits per second
over the channel which has a two millisecond transmission delay.

InternetStackHelper
We now have nodes and devices configured, but we don't have any protocol stacks installed on our nodes. The next two lines of code will take care of that.

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

The InternetStackHelper is a topology helper that is to internet stacks what the PointToPointHelper is to point-to-point net devices.

The Install method takes a NodeContainer as a parameter. When it is executed, it will install an Internet Stack(TCP, UDP, IP, etc.) on each of the nodes in the node container.

Ipv4AddressHelper
Next we need to associate the devices on our nodes with IP addresses. We provide a topology helper to manage the allocation of IP addresses. The only user-visible API is to set the base IP address and network mask to use when performing the actual address allocation (which is done at a lower level inside the helper).

The next two lines of code in our example script, first.cc,

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

declare an address helper object and tell it that it should begin allocating IP addresses from the network 10.1.1.0 using the mask 255.255.255.0 to define the allocatable bits.

By default the addresses allocated will start at one and increase monotonically, so the first address allocated from this base will be 10.1.1.1, followed by 10.1.1.2, etc. The low level |ns3| system actually remembers all of the IP addresses allocated and will generate a fatal error if you accidentally cause the same address
to be generated twice (which is a very hard to debug error, by the way).

The next line of code,

```
Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

performs the actual address assignment.

In |ns3| we make the association between an IP address and a device using an Ipv4Interface object.

Just as we sometimes need a list of net devices created by a helper for future reference we sometimes need a list of Ipv4Interface objects.

The Ipv4InterfaceContainer provides this functionality.

Now we have a point-to-point network built, with stacks installed and IP addresses assigned. What we need at this point are applications to generate traffic.

Applications
Another one of the core abstractions of the ns-3 system is the Application. In this script we use two specializations of the core |ns3| class Application called UdpEchoServerApplication and UdpEchoClientApplication.

Just as we have in our previous explanations, we use helper objects to help configure and manage the underlying objects.

Here, we use UdpEchoServerHelper and UdpEchoClientHelper objects to make our lives easier.

UdpEchoServerHelper
The following lines of code in our example script, first.cc, are used to set up a UDP echo server application on one of the nodes we have previously created.

```
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install(nodes.Get (1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
```

The first line of code in the above snippet declares the UdpEchoServerHelper.

As usual, this isn't the application itself, it is an object used to help us create the actual applications. One of our conventions is to place required Attributes in the helper constructor.

In this case, the helper can't do anything useful unless it is provided with a port number that the client also knows about. Rather than just picking one and hoping it all works out, we require the port number as a parameter to the constructor.

The constructor, in turn, simply does a SetAttribute with the passed value.

If you want, you can set the "Port" Attribute to another value later using SetAttribute.

Similar to many other helper objects, the UdpEchoServerHelper object has an Install method.
It is the execution of this method that actually causes the underlying echo server application to be instantiated and attached to a node.

Interestingly, the Install method takes a NodeContainter as a parameter just as the other Install methods we have seen.

This is actually what is passed to the method even though it doesn't look so in this case.

There is a C++ implicit conversion at work here that takes the result of nodes.Get (1) (which returns a smart pointer to a node object --- Ptr<Node>) and uses that in a constructor for an unnamed NodeContainer that is then passed to Install.

If you are ever at a loss to find a particular method signature in C++ code that compiles and runs just fine, look for these kinds of implicit conversions.

We now see that echoServer.Install is going to install a UdpEchoServerApplication on the node found at index number one of the NodeContainer we used to manage our nodes.

Install will return a container that holds pointers to all of the applications (one in this case since we passed a NodeContainer containing one node) created by the helper.

Applications require a time to "start" generating traffic and may take an optional time to "stop". We provide both.

These times are set using the ApplicationContainer methods Start and Stop.

These methods take Time parameters.

In this case, we use an explicit C++ conversion sequence to take the C++ double 1.0 and convert it to an |ns3| Time object using a Seconds cast.

Be aware that the conversion rules may be controlled by the model author, and C++ has its own rules, so you can't always just assume that parameters will be happily converted for you.

The two lines,

```
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
```

will cause the echo server application to Start (enable itself) at one second into the simulation and to Stop (disable itself) at ten seconds into the simulation.

By virtue of the fact that we have declared a simulation event (the application stop event) to be executed at ten seconds, the simulation will last at least ten seconds.

UdpEchoClientHelper

The echo client application is set up in a method substantially similar to that for the server. There is an underlying UdpEchoClientApplication that is managed by an UdpEchoClientHelper.

```
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));
```

For the echo client, however, we need to set five different Attributes. The first two Attributes are set during construction of the UdpEchoClientHelper.

We pass parameters that are used (internally to the helper) to set the "RemoteAddress" and "RemotePort" Attributes in accordance with our convention to make required Attributes parameters in the helper constructors.

Recall that we used an Ipv4InterfaceContainer to keep track of the IP addresses we assigned to our devices.

The zeroth interface in the interfaces container is going to correspond to the IP address of the zeroth node in the nodes container.

The first interface in the interfaces container corresponds to the IP address of the first node in the nodes container. So, in the first line of code (from above), we are creating the helper and telling it so set the remote address of the client to be the IP address assigned to the node on which the server resides. We also tell it to arrange to send packets to port nine.

The "MaxPackets" Attribute tells the client the maximum number of packets we allow it to send during the simulation.

The "Interval" Attribute tells the client how long to wait between packets, and the "PacketSize" Attribute tells the client how large its packet payloads should be.

With this particular combination of Attributes, we are telling the client to send one 1024-byte packet.

Just as in the case of the echo server, we tell the echo client to Start and Stop, but here we start the client one second after the server is enabled (at two seconds into the simulation).

Simulator
What we need to do at this point is to actually run the simulation. This is done using the global function Simulator::Run.

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

When we previously called the methods,
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
...
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

we actually scheduled events in the simulator at 1.0 seconds, 2.0 seconds and two events at 10.0 seconds.

When Simulator::Run is called, the system will begin looking through the list of scheduled events and executing them.

First it will run the event at 1.0 seconds, which will enable the echo 18server application (this event may, in turn, schedule many other events).

Then it will run the event scheduled for t=2.0 seconds which will start the echo client application. Again, this event may schedule many more events.

The start event implementation in the echo client application will begin the data transfer phase of the simulation by sending a packet to the server.

The act of sending the packet to the server will trigger a chain of events that will be automatically scheduled behind the scenes and which will perform the mechanics of the packet echo according to the various timing parameters that we have set in

the script.

Eventually, since we only send one packet (recall the MaxPackets Attribute was set to one), the chain of events triggered by that single client echo request will taper off and the simulation will go idle. Once this happens, the remaining events will be the Stop events for the server and the client.

When these events are executed, there are no further events to process and Simulator::Run returns.

The simulation is then complete.

All that remains is to clean up. This is done by calling the global function Simulator::Destroy.

As the helper functions (or low level |ns3| code) executed, they arranged it so that hooks were inserted in the simulator to destroy all of the objects that were created. You did not have to keep track of any of these objects yourself --- all you had to do was to call Simulator::Destroy and exit.

The |ns3| system took care of the hard part for you.
The remaining lines of our first |ns3| script, first.cc, do just that:

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

When the simulator will stop?
|ns3| is a Discrete Event (DE) simulator. In such a simulator, each event is associated with its execution time, and the simulation proceeds by executing events in the temporal order of simulation time. Events may cause future events to be 19scheduled (for example, a timer may reschedule itself to expire at the next interval).

The initial events are usually triggered by each object, e.g., Ipv6 will schedule Router Advertisements, Neighbor Solicitations, etc., an Application schedule the first packet sending event, etc.

When an event is processed, it may generate zero, one or more events. As a simulation executes, events are consumed, but more events may (or may not) be generated.

The simulation will stop automatically when no further events are in the event queue, or when a special Stop event is found. The Stop event is created through the Simulator::Stop (stopTime); function.

There is a typical case where Simulator::Stop is absolutely necessary to stop the simulation: when there is a self-sustaining event. Self-sustaining (or recurring) events are events that always reschedule themselves. As a consequence, they always keep the event queue non-empty.

There are many protocols and modules containing recurring
events, e.g.:

FlowMonitor - periodic check for lost packets

RIPng - periodic broadcast of routing tables update etc.

In these cases, Simulator::Stop is necessary to gracefully stop the simulation.

In addition, when |ns3| is in emulation mode, the RealtimeSimulator is used to keep the simulation clock aligned with the machine clock, and Simulator::Stop is necessary to stop the process.

Many of the simulation programs in the tutorial do not explicitly call Simulator::Stop, since the event queue will automatically run out of events. However, these programs will also accept a call to Simulator::Stop.

For example, the following additional statement in the first example program will schedule an explicit stop at 11 seconds:

```
+ Simulator::Stop (Seconds (11.0));
Simulator::Run ();
Simulator::Destroy ();
return 0;
}
```

The above will not actually change the behavior of this program, since this particular simulation naturally ends after 10 seconds.
But if you were to change the stop time in the above statement from 11 seconds to 1 second, you would notice that the simulation stops before any output is printed to the screen (since the output occurs around time 2 seconds of simulation time).

It is important to call Simulator::Stop before calling Simulator::Run;

otherwise, Simulator::Run may never return control to the main program to execute the stop!

Building Your Script
We have made it trivial to build your simple scripts. All you have to do is to drop your script into the scratch directory and it will automatically be built if you run Waf.

Let's try it.

Copy examples/tutorial/first.cc into the scratch directory after changing back into the top level directory.

```
$ cd ../..
$ cp examples/tutorial/first.cc scratch/myfirst.cc
```

Now build your first example script using waf:
```
$ ./waf
```

You should see messages reporting that your myfirst example was built successfully.

```
Waf: Entering directory `/home/craigdo/repos/ns-3-allinone/ns-3-dev/build'
[614/708] cxx: scratch/myfirst.cc ->

build/debug/scratch/myfirst_3.o
[706/708] cxx_link: build/debug/scratch/myfirst_3.o ->
build/debug/scratch/myfirst

Waf: Leaving directory `/home/craigdo/repos/ns-3-allinone/ns-3-dev/build'

'build' finished successfully (2.357s)
```

You can now run the example (note that if you build your program in the scratch directory you must run it out of the scratch directory):

$ ./waf --run scratch/myfirst

You should see some output:
```
Waf: Entering directory `/home/craigdo/repos/ns-3-allinone/ns-3-dev/build'

Waf: Leaving directory `/home/craigdo/repos/ns-3-allinone/ns-3-dev/build'
'build' finished successfully (0.418s)

Sent 1024 bytes to 10.1.1.2
Received 1024 bytes from 10.1.1.1
Received 1024 bytes from 10.1.1.2
```

Here you see that the build system checks to make sure that the file has been build and then runs it.
You see the logging component on the echo client indicate that it has sent one 1024 byte packet to the Echo Server on 10.1.1.2.

You also see the logging component on the echo server say that it has received the 1024 bytes from 10.1.1.1.

The echo server silently echoes the packet and you see the echo client log that it has received its packet back from the server.

Practical 6: **Experiment Specific Instructions**

Create two nodes: an access point (AP) and a base station (BS). The BS will sends message to the AP, also the AP will sends a response back to the base station. Since the medium of communication is wireless (over air), the BS could send (or receive) messages to (or from) the AP only when ir is within a certain distance from the AP.

1. Create the nodes and hold them in a container:

```
NodeContainer wifiStaNodes, wifiApNode;
wifiStaNodes.Create (nWifi);
wifiApNode = wifiStaNodes.Get (0);
```

2.Create channel for communication:

```
YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();

phy.SetChannel (channel.Create ());
WifiHelper wifi = WifiHelper::Default ();

wifi.SetRemoteStationManager ("ns3::AarfWifiManager");
NqosWifiMacHelper mac = NqosWifiMacHelper::Default ();
```

3. Set up MAC for base station

```
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac","Ssid", SsidValue (ssid),"ActiveProbing",
BooleanValue (false));

NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes.Get (1));
```

4. Set up MAC for AP:

```
mac.SetType ("ns3::ApWifiMac","Ssid",
SsidValue(ssid),"BeaconGeneration",BooleanValue
(true),"BeaconInterval",TimeValue (Seconds (5)));


NetDeviceContainer apDevice;
apDevice = wifi.Install (phy, mac, wifiApNode);
```

5. Set mobility of the nodes:

```
MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                               "MinX", DoubleValue (0.0),

"MinY", DoubleValue (0.0),"DeltaX", DoubleValue (xDistance),"DeltaY",
DoubleValue (10.0),"GridWidth", UintegerValue (3),"LayoutType",
StringValue"RowFirst"));
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiStaNodes);
```

6.Add Internet layers stack:

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

7. Assign IP address to each device:

```
Ipv4AddressHelper address;
Ipv4InterfaceContainer wifiInterfaces, wifiApInterface;

address.SetBase(10.1.1.0","255.255.255.0");
wifiApInterface = address.Assign (apDevice);
wifiInterfaces = address.Assign (staDevices);
```

8.Create and setup applications (traffic sink):

```
UdpEchoServerHelper echoServer (9);// Port # 9
ApplicationContainer serverApps = echoServer.Install (wifiApNode);

serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (4.0));
```

9. Create and setup applications (traffic source):

```
UdpEchoClientHelper echoClient (wifiApInterface.GetAddress (0), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.)));
echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
ApplicationContainer clientApps = echoClient.Install (wifiStaNodes.Get (1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (3.0));
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
Simulator::Stop (Seconds (4.0));
```

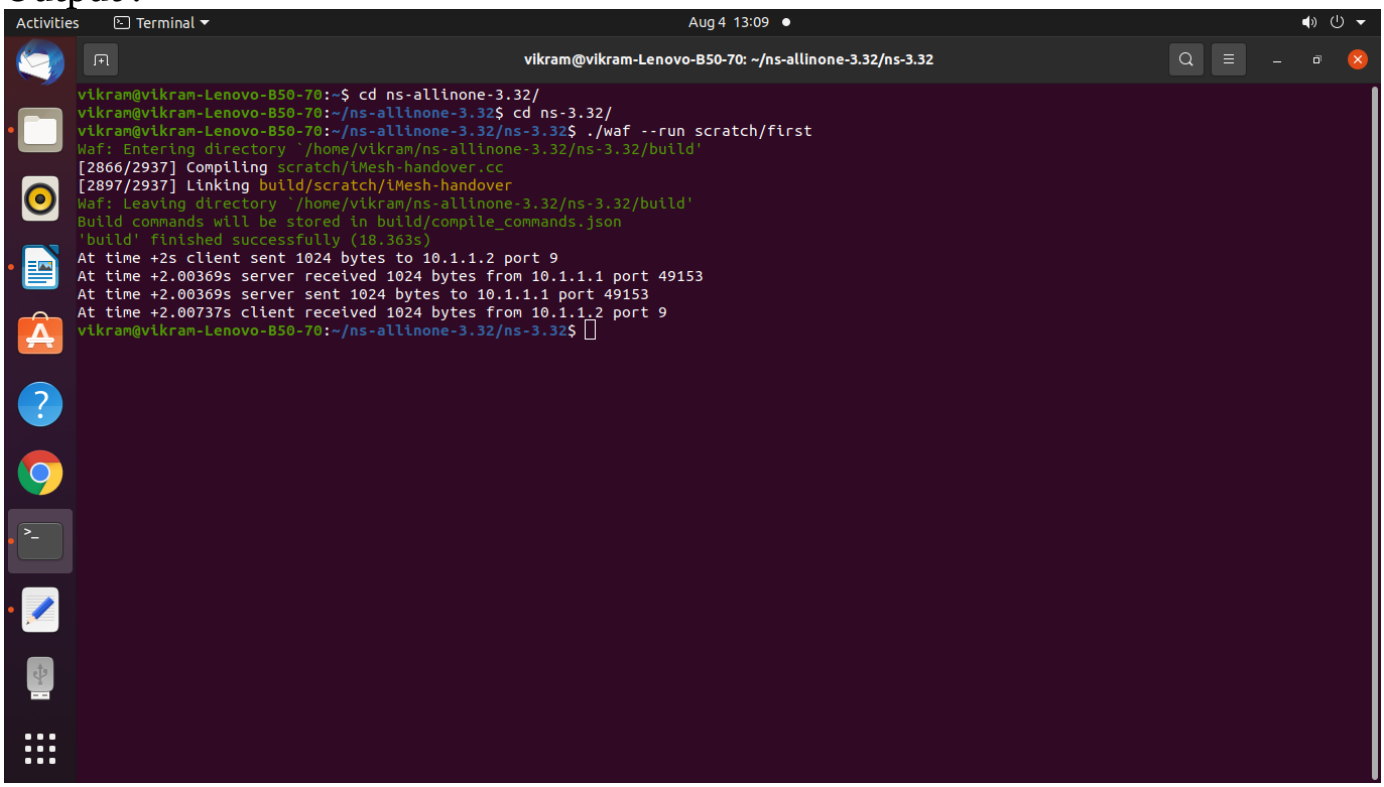
Practical No. 7 : Defualt Network Topology (Point-to-Point)

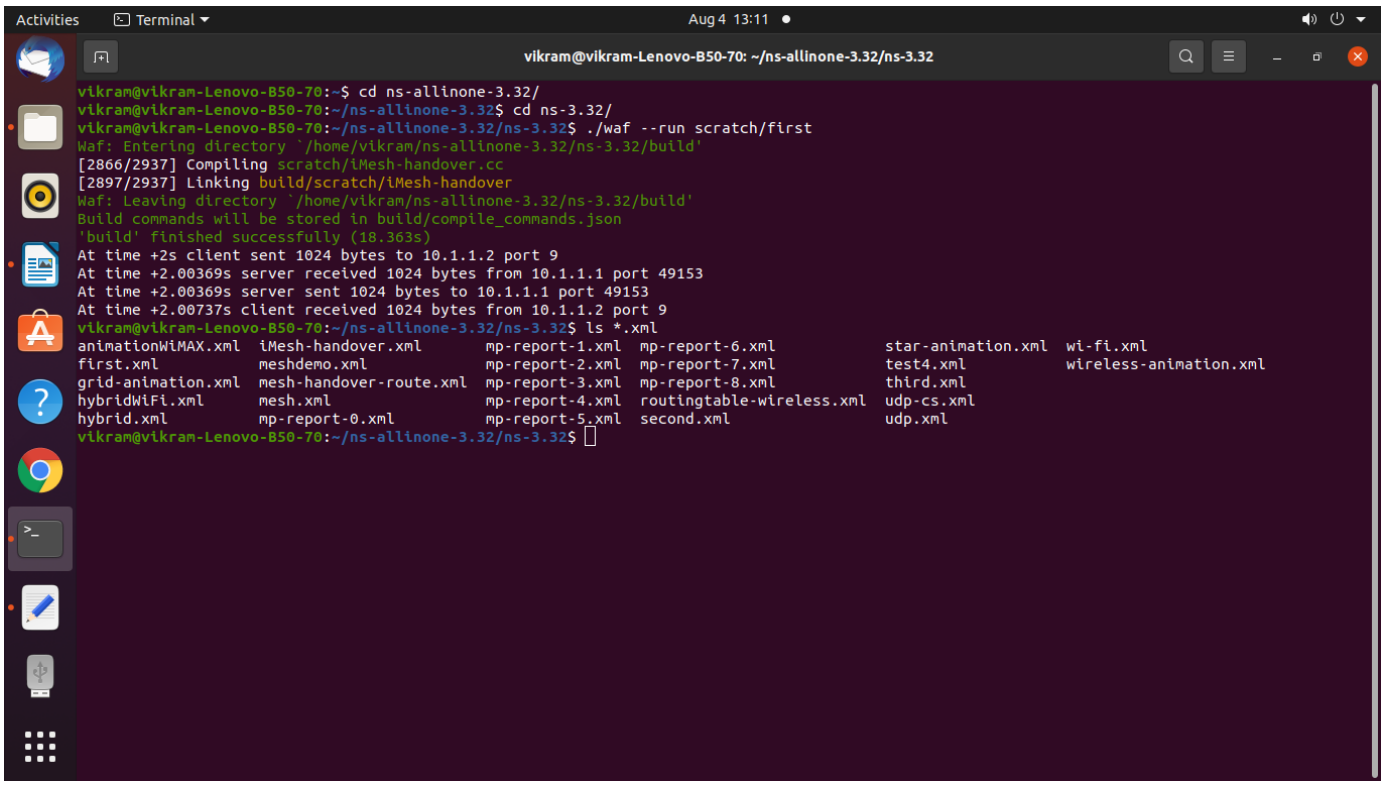
```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the 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"

//netAnimation

#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__);
```

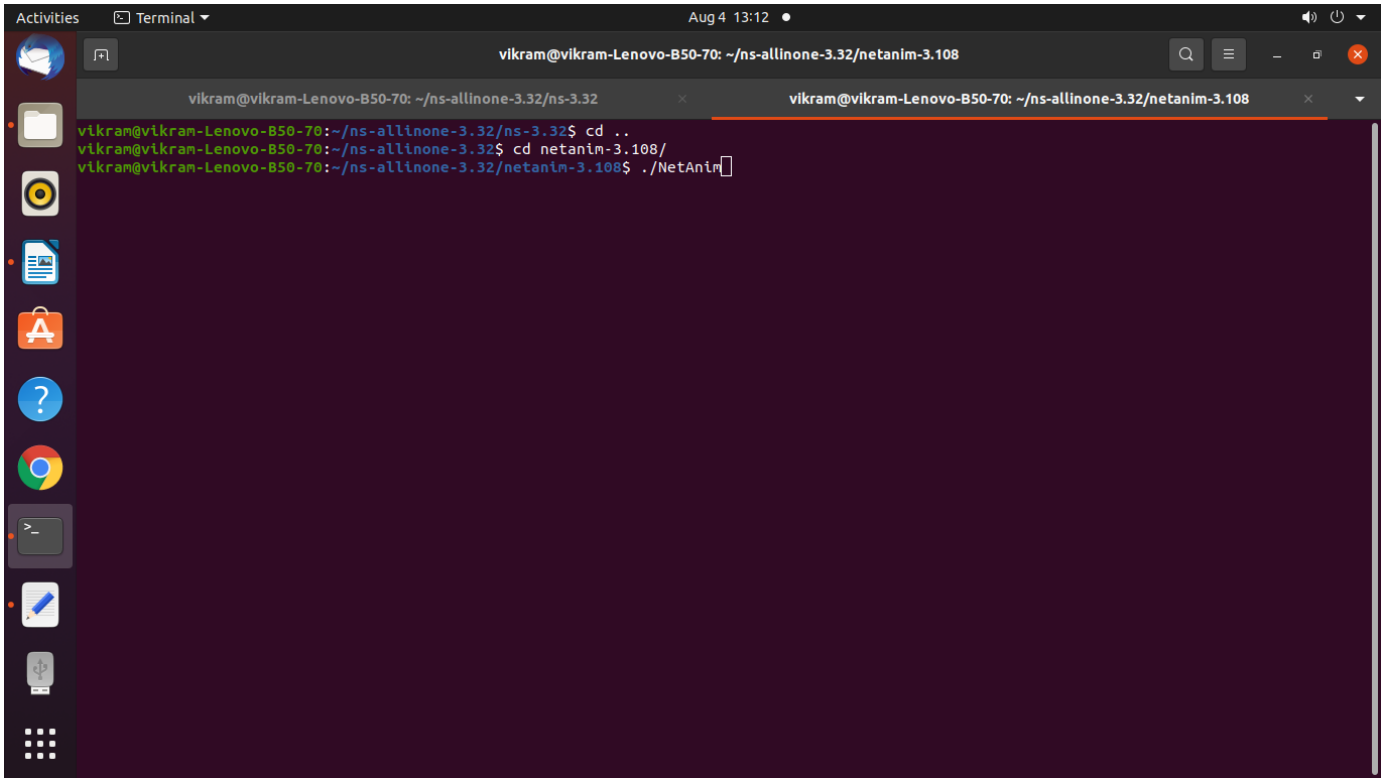
```
  cmd.Parse (argc, argv);

  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 (1024));

  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);
```

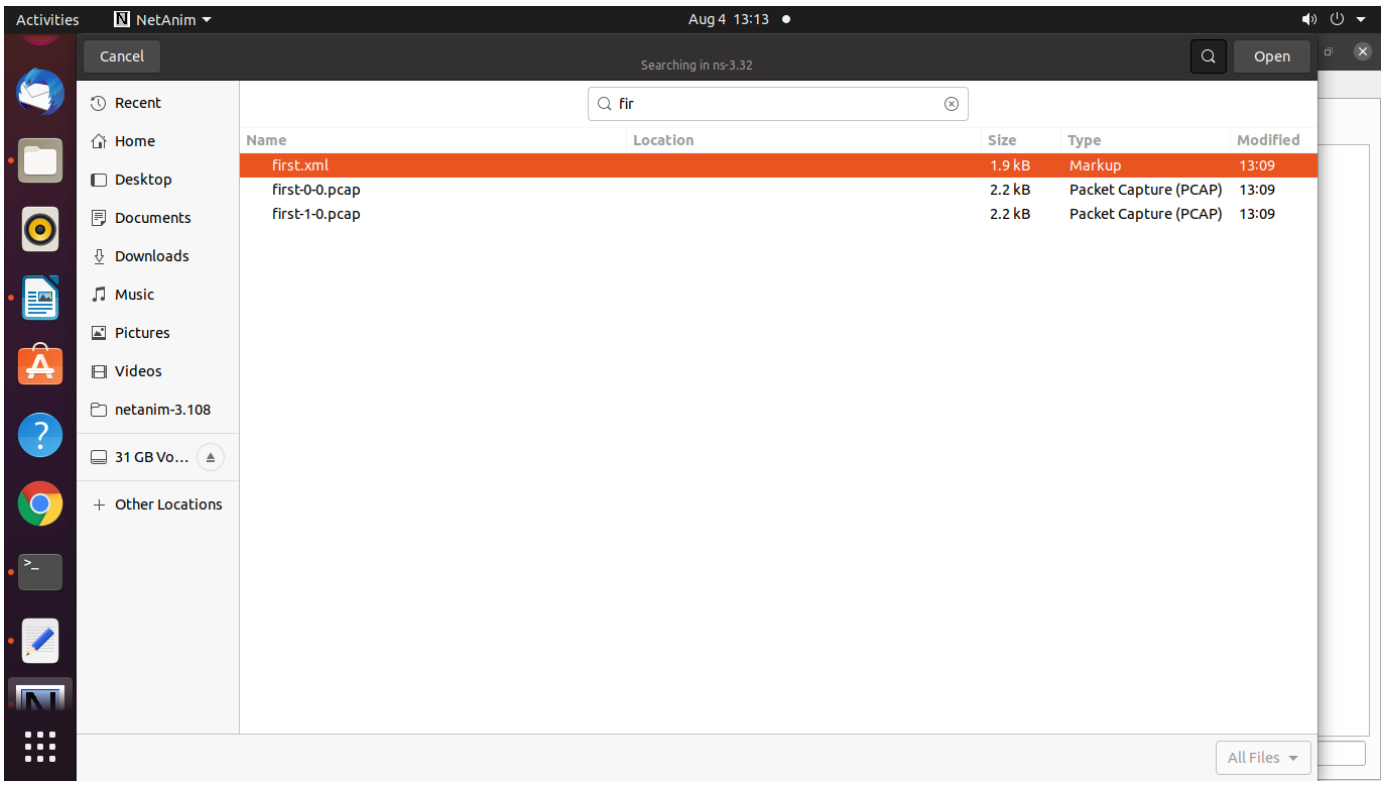
```
anim.EnablePacketMetadata(true);


pointToPoint.EnablePcapAll("first");

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

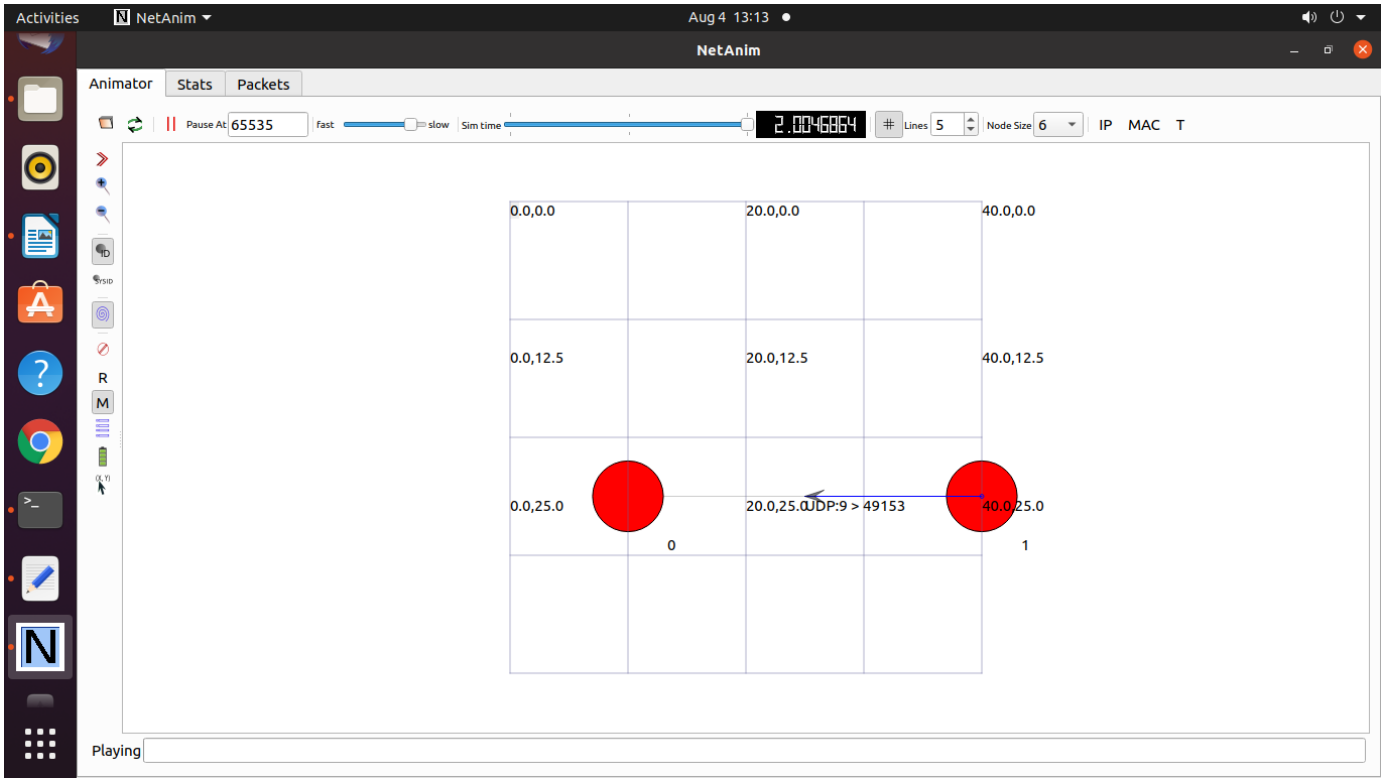
Output :

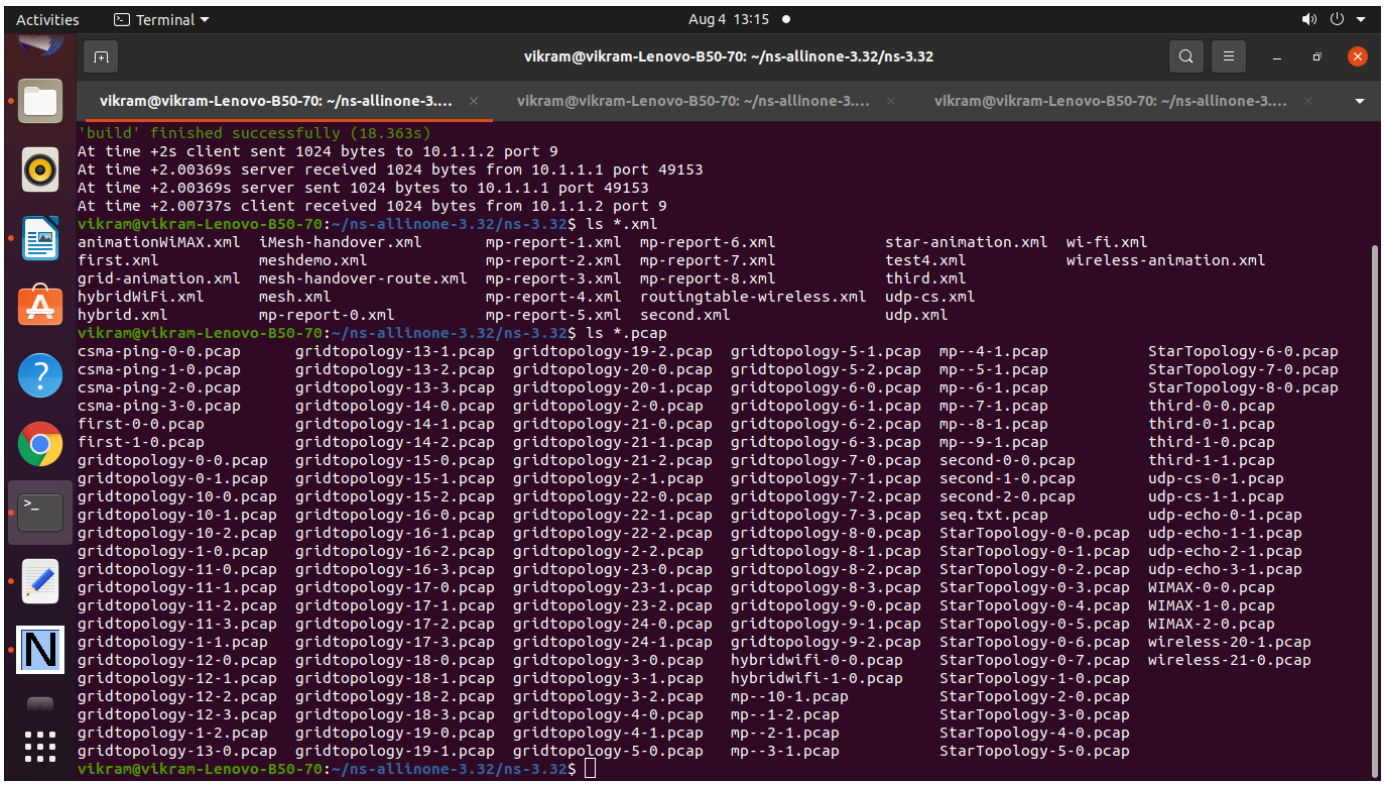


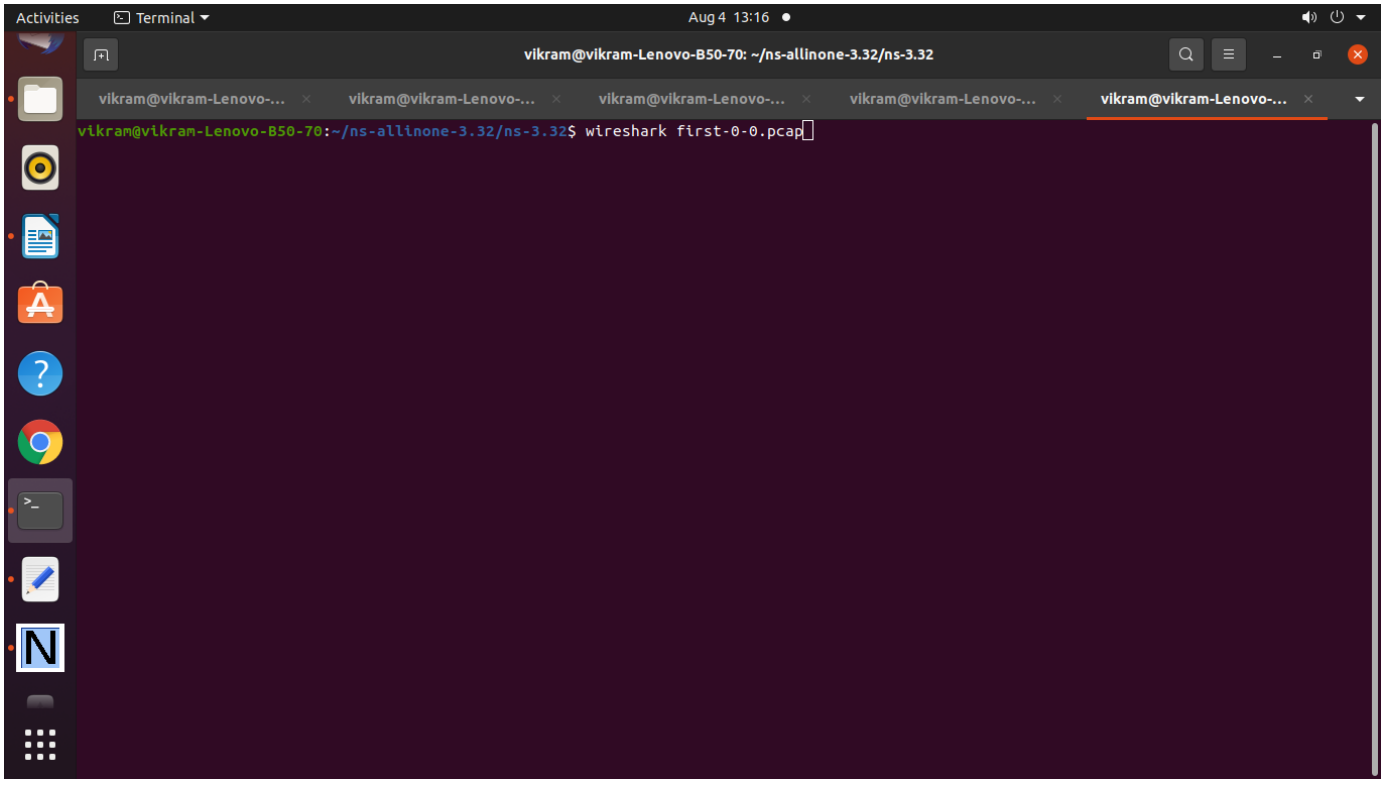

Activities
Terminal
Aug 4 13:11
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70:~$ cd ns-allinone-3.32/
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32$ cd ns-3.32/
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/first
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
[2866/2937] Compiling scratch/iMesh-handover.cc
[2897/2937] Linking build/scratch/iMesh-handover
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (18.363s)
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
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ls *.xml
animationWiMAX.xml iMesh-handover.xml mp-report-1.xml mp-report-6.xml star-animation.xml wi-fi.xml
first.xml meshdemo.xml mp-report-2.xml mp-report-7.xml test4.xml wireless-animation.xml
grid-animation.xml mesh-handover-route.xml mp-report-3.xml mp-report-8.xml third.xml
hybridWiFi.xml mesh.xml mp-report-4.xml routingtable-wireless.xml udp-cs.xml
hybrid.xml mp-report-0.xml mp-report-5.xml second.xml udp.xml
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$

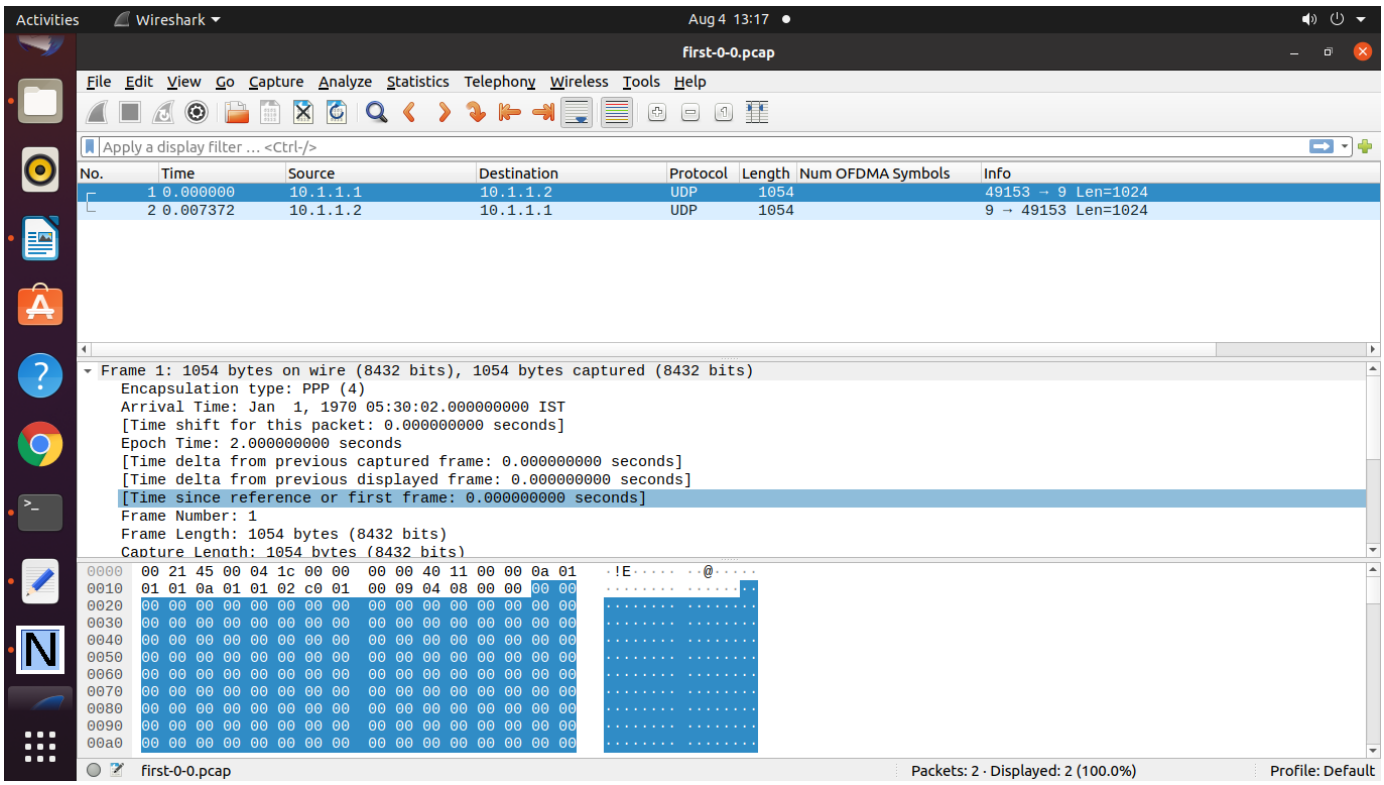

Activities
Terminal
Aug 4 13:12
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/netanim-3.108
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/netanim-3.108
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ cd ..
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32$ cd netanim-3.108/
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/netanim-3.108$ ./NetAnim


Activities
NetAnim
Aug 4 13:13
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
31 GB Vo...
Other Locations
fir
Name
Location
Size
Type
Modified
first.xml
1.9 kB
Markup
13:09
first-0-0.pcap
2.2 kB
Packet Capture (PCAP)
13:09
first-1-0.pcap
2.2 kB
Packet Capture (PCAP)
13:09
All Files


Activities
NetAnim
Aug 4 13:13
NetAnim
Animator
Stats
Packets
Pause At
65535
fast
slow
Sim time
2.004684
Lines
5
Node Size
6
IP
MAC
T
0.0,0.0
20.0,0.0
40.0,0.0
0.0,12.5
20.0,12.5
40.0,12.5
0.0,25.0
20.0,25.0
UDP:9 > 49153
40.0,25.0
0
1
Playing

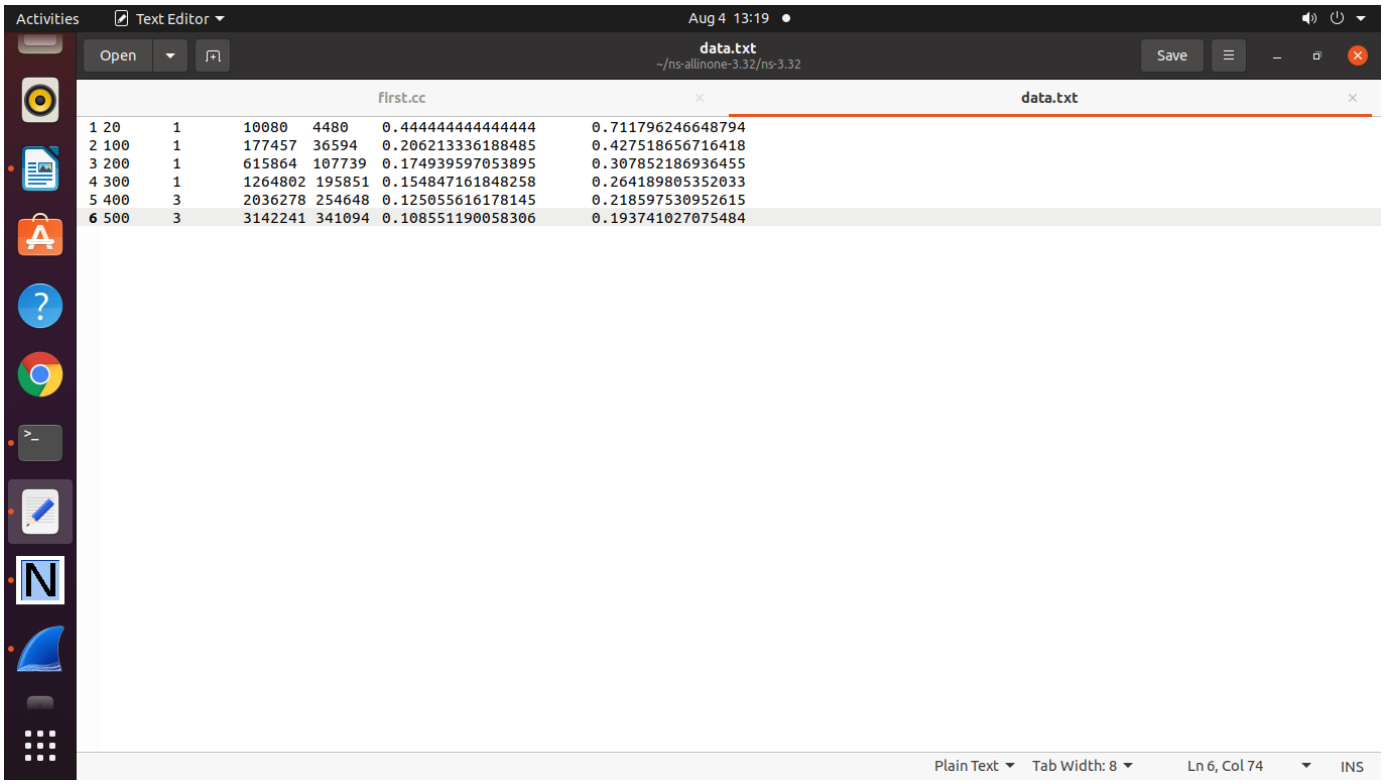

Activities
Terminal
Aug 4 13:15
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
'build' finished successfully (18.363s)
At time +2s client sent 1024 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
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ls *.xml
animationWiMAX.xml iMesh-handover.xml mp-report-1.xml mp-report-6.xml star-animation.xml wi-fi.xml
first.xml meshdemo.xml mp-report-2.xml mp-report-7.xml test4.xml wireless-animation.xml
grid-animation.xml mesh-handover-route.xml mp-report-3.xml mp-report-8.xml third.xml
hybridWiFi.xml mesh.xml mp-report-4.xml routingtable-wireless.xml udp-cs.xml
hybrid.xml mp-report-0.xml mp-report-5.xml second.xml udp.xml
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ls *.pcap
csma-ping-0-0.pcap gridtopology-13-1.pcap gridtopology-19-2.pcap gridtopology-5-1.pcap mp--4-1.pcap StarTopology-6-0.pcap
csma-ping-1-0.pcap gridtopology-13-2.pcap gridtopology-20-0.pcap gridtopology-5-2.pcap mp--5-1.pcap StarTopology-7-0.pcap
csma-ping-2-0.pcap gridtopology-13-3.pcap gridtopology-20-1.pcap gridtopology-6-0.pcap mp--6-1.pcap StarTopology-8-0.pcap
csma-ping-3-0.pcap gridtopology-14-0.pcap gridtopology-2-0.pcap gridtopology-6-1.pcap mp--7-1.pcap third-0-0.pcap
first-0-0.pcap gridtopology-14-1.pcap gridtopology-21-0.pcap gridtopology-6-2.pcap mp--8-1.pcap third-0-1.pcap
first-1-0.pcap gridtopology-14-2.pcap gridtopology-21-1.pcap gridtopology-6-3.pcap mp--9-1.pcap third-1-0.pcap
gridtopology-0-0.pcap gridtopology-15-0.pcap gridtopology-21-2.pcap gridtopology-7-0.pcap second-0-0.pcap third-1-1.pcap
gridtopology-0-1.pcap gridtopology-15-1.pcap gridtopology-2-1.pcap gridtopology-7-1.pcap second-1-0.pcap udp-cs-0-1.pcap
gridtopology-10-0.pcap gridtopology-15-2.pcap gridtopology-22-0.pcap gridtopology-7-2.pcap second-2-0.pcap udp-cs-1-1.pcap
gridtopology-10-1.pcap gridtopology-16-0.pcap gridtopology-22-1.pcap gridtopology-7-3.pcap seq.txt.pcap udp-echo-0-1.pcap
gridtopology-10-2.pcap gridtopology-16-1.pcap gridtopology-22-2.pcap gridtopology-8-0.pcap StarTopology-0-0.pcap udp-echo-1-1.pcap
gridtopology-1-0.pcap gridtopology-16-2.pcap gridtopology-2-2.pcap gridtopology-8-1.pcap StarTopology-0-1.pcap udp-echo-2-1.pcap
gridtopology-11-0.pcap gridtopology-16-3.pcap gridtopology-23-0.pcap gridtopology-8-2.pcap StarTopology-0-2.pcap udp-echo-3-1.pcap
gridtopology-11-1.pcap gridtopology-17-0.pcap gridtopology-23-1.pcap gridtopology-8-3.pcap StarTopology-0-3.pcap WIMAX-0-0.pcap
gridtopology-11-2.pcap gridtopology-17-1.pcap gridtopology-23-2.pcap gridtopology-9-0.pcap StarTopology-0-4.pcap WIMAX-1-0.pcap
gridtopology-11-3.pcap gridtopology-17-2.pcap gridtopology-24-0.pcap gridtopology-9-1.pcap StarTopology-0-5.pcap WIMAX-2-0.pcap
gridtopology-1-1.pcap gridtopology-17-3.pcap gridtopology-24-1.pcap gridtopology-9-2.pcap StarTopology-0-6.pcap wireless-20-1.pcap
gridtopology-12-0.pcap gridtopology-18-0.pcap gridtopology-3-0.pcap hybridwifi-0-0.pcap StarTopology-0-7.pcap wireless-21-0.pcap
gridtopology-12-1.pcap gridtopology-18-1.pcap gridtopology-3-1.pcap hybridwifi-1-0.pcap StarTopology-1-0.pcap
gridtopology-12-2.pcap gridtopology-18-2.pcap gridtopology-3-2.pcap mp--10-1.pcap StarTopology-2-0.pcap
gridtopology-12-3.pcap gridtopology-18-3.pcap gridtopology-4-0.pcap mp--1-2.pcap StarTopology-3-0.pcap
gridtopology-1-2.pcap gridtopology-19-0.pcap gridtopology-4-1.pcap mp--2-1.pcap StarTopology-4-0.pcap
gridtopology-13-0.pcap gridtopology-19-1.pcap gridtopology-5-0.pcap mp--3-1.pcap StarTopology-5-0.pcap
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$


Activities
Terminal
Aug 4 13:16
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ wireshark first-0-0.pcap



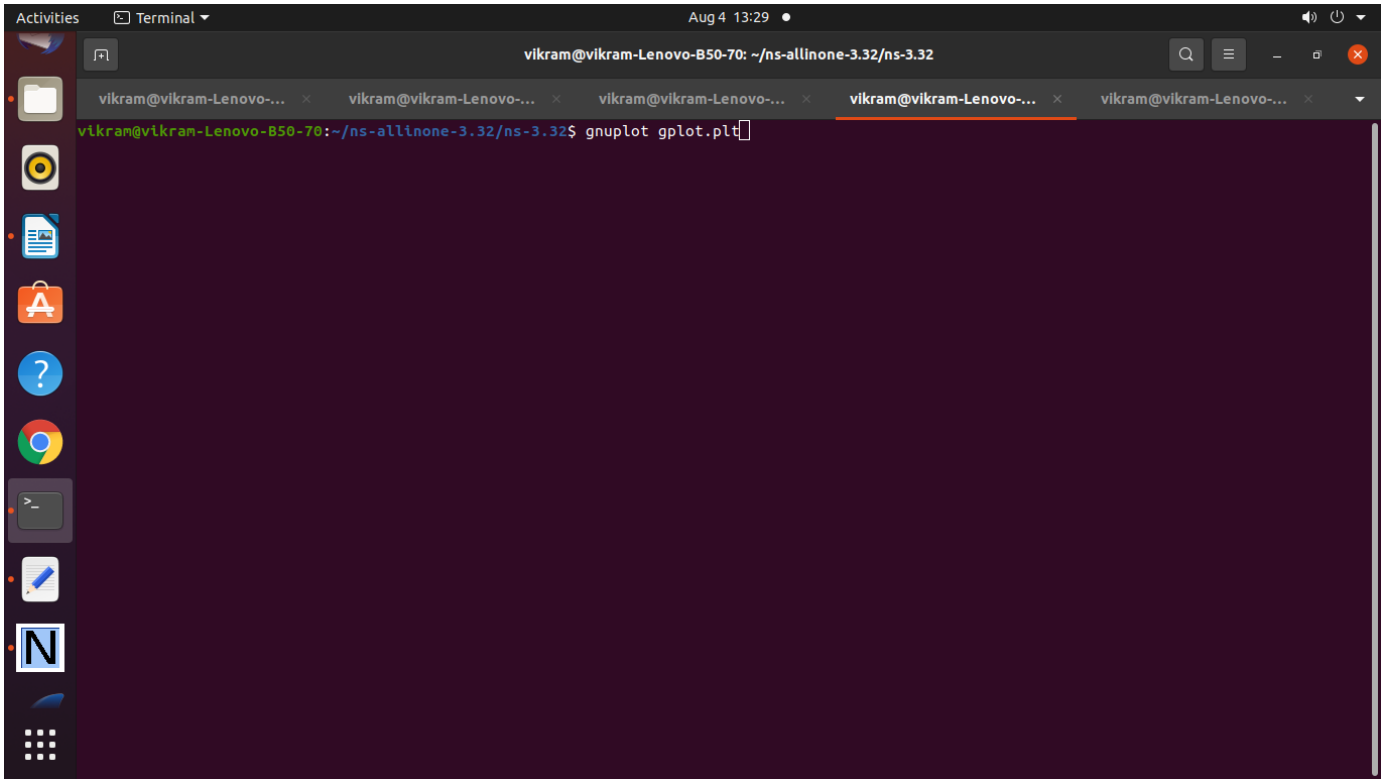
To Generate the Graph refer data.txt file

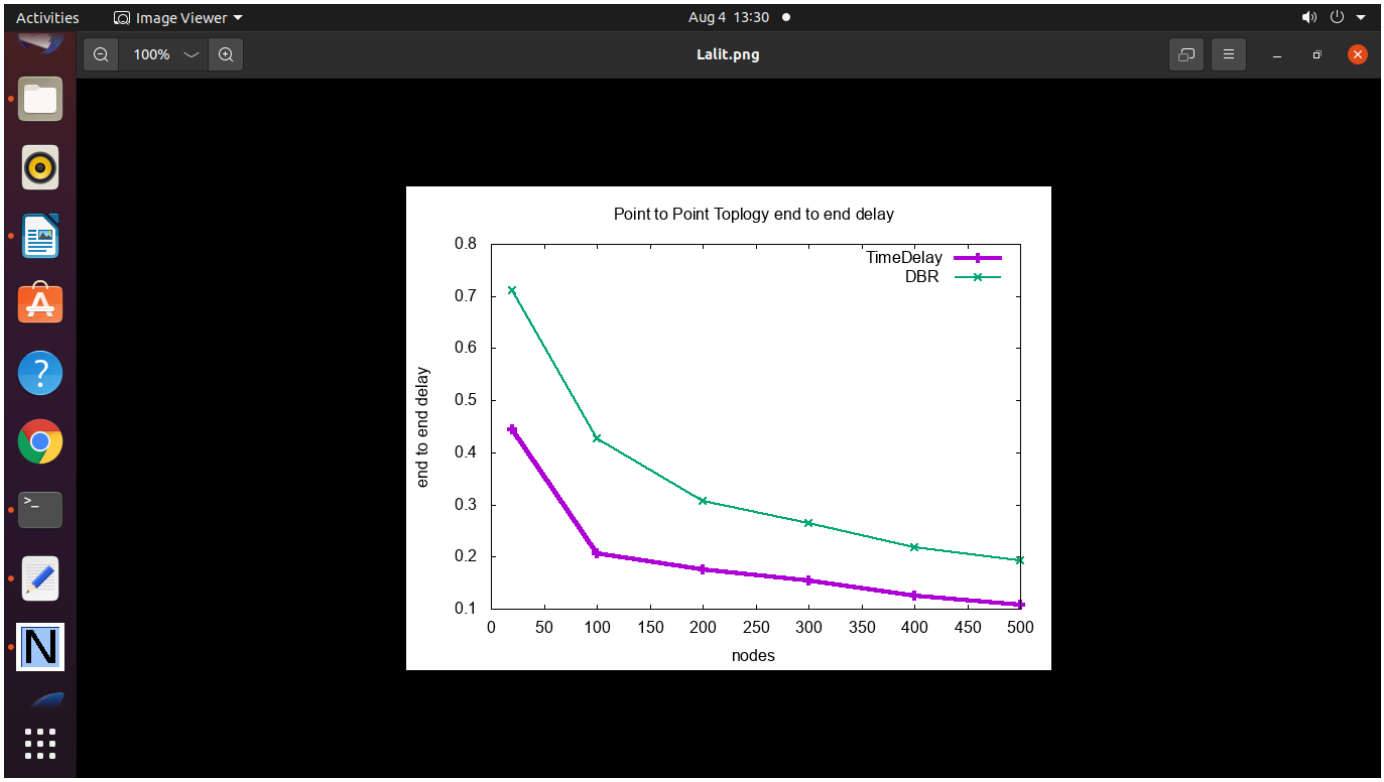
20	1	10080	4480	0.444444444444444	0.711796246648794
100	1	177457	36594	0.206213336188485	0.427518656716418
200	1	615864	107739	0.174939597053895	0.307852186936455
300	1	1264802	195851	0.154847161848258	0.264189805352033
400	3	2036278	254648	0.125055616178145	0.218597530952615
500	3	3142241	341094	0.108551190058306	0.193741027075484



gplot.plt file to Generate Graph

set terminal png
set output "Lalit.png"
set title "Point to Point Toplogy end to end delay"
set xlabel "nodes"
set ylabel "end to end delay"
plot "data.txt" using 1: 5 with linespoint title "TimeDelay" lw 4,"data.txt" using 1: 6 with linespoint title "DBR " lw 2


Activities
Terminal
Aug 4 13:29
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ gnuplot gplot.plt


Activities
Image Viewer
Aug 4 13:30
100%
Lalit.png
Point to Point Toplogy end to end delay
TimeDelay
DBR
end to end delay
nodes
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
300
350
400
450
500

==========

Practical 8: **Program to simulate Hybrid Topology (Point-to-Point + Bus Toplogy)**

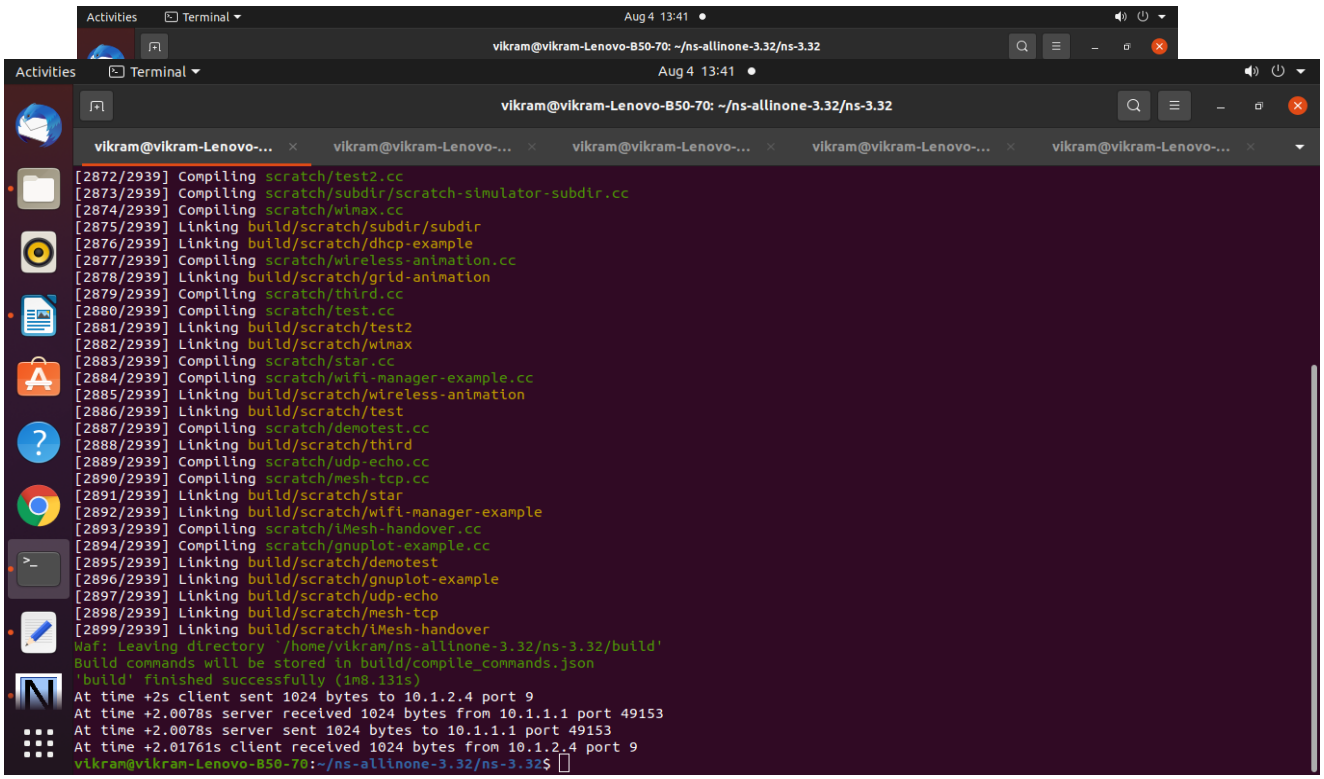
```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the 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"

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

//netAnimation

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


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");
```

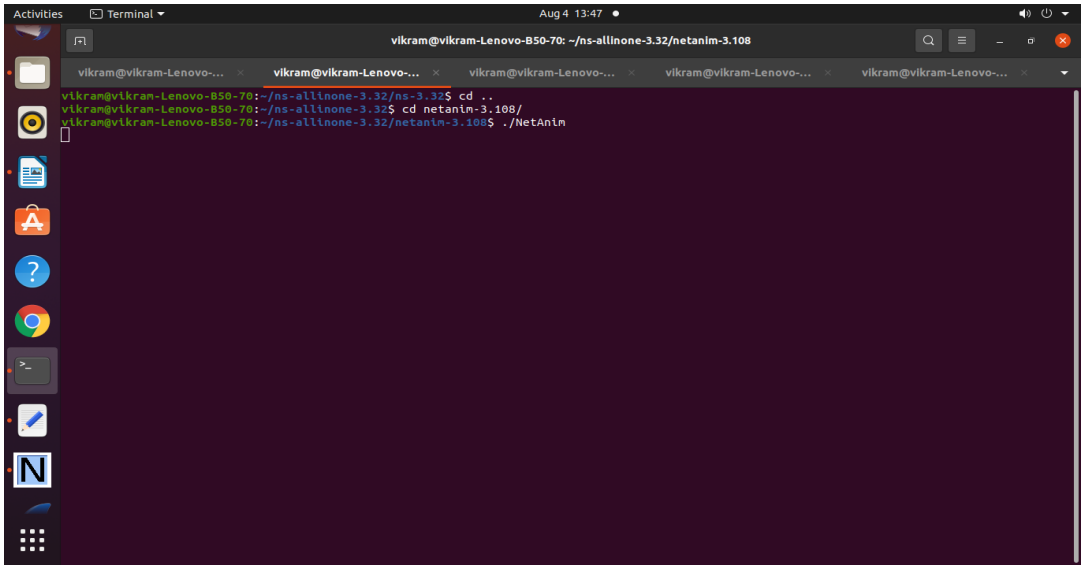
```
int
main (int argc, char *argv[])
{
  bool verbose = true;
  uint32_t nCsma = 3;

  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);
```

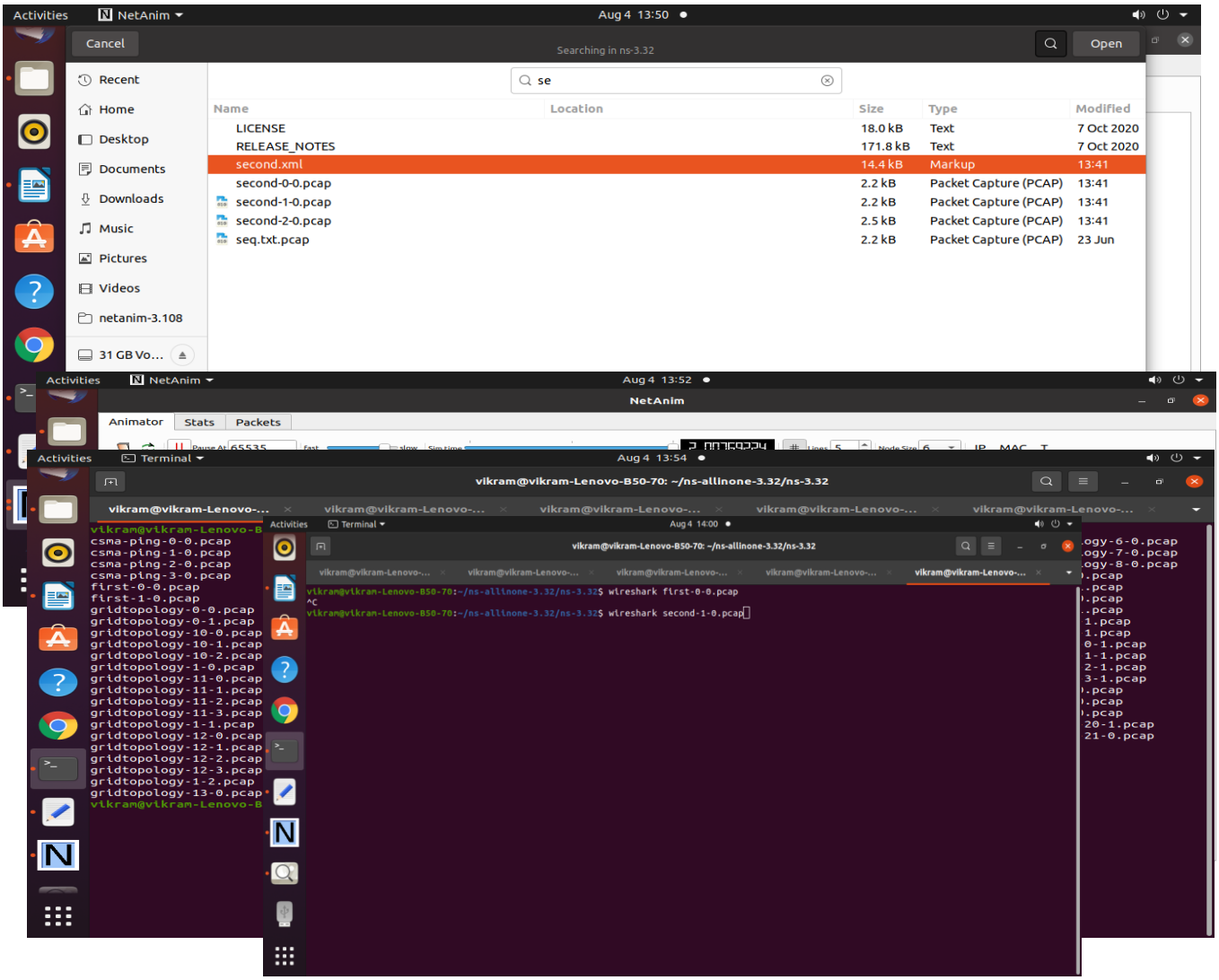
```
  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces;
  p2pInterfaces = address.Assign (p2pDevices);

  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);

  //For Net Anim
MobilityHelper mobility;
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(p2pNodes);
mobility.Install(csmaNodes);
AnimationInterface anim("second.xml");
AnimationInterface::SetConstantPosition (p2pNodes.Get(0), 10, 25);
AnimationInterface ::SetConstantPosition(p2pNodes.Get(1), 20,25);
AnimationInterface ::SetConstantPosition(csmaNodes.Get(1),40,25);
AnimationInterface ::SetConstantPosition(csmaNodes.Get(2),50,25);
AnimationInterface ::SetConstantPosition(csmaNodes.Get(3),60,25);
anim.EnablePacketMetadata(true);


  Simulator::Run ();
```

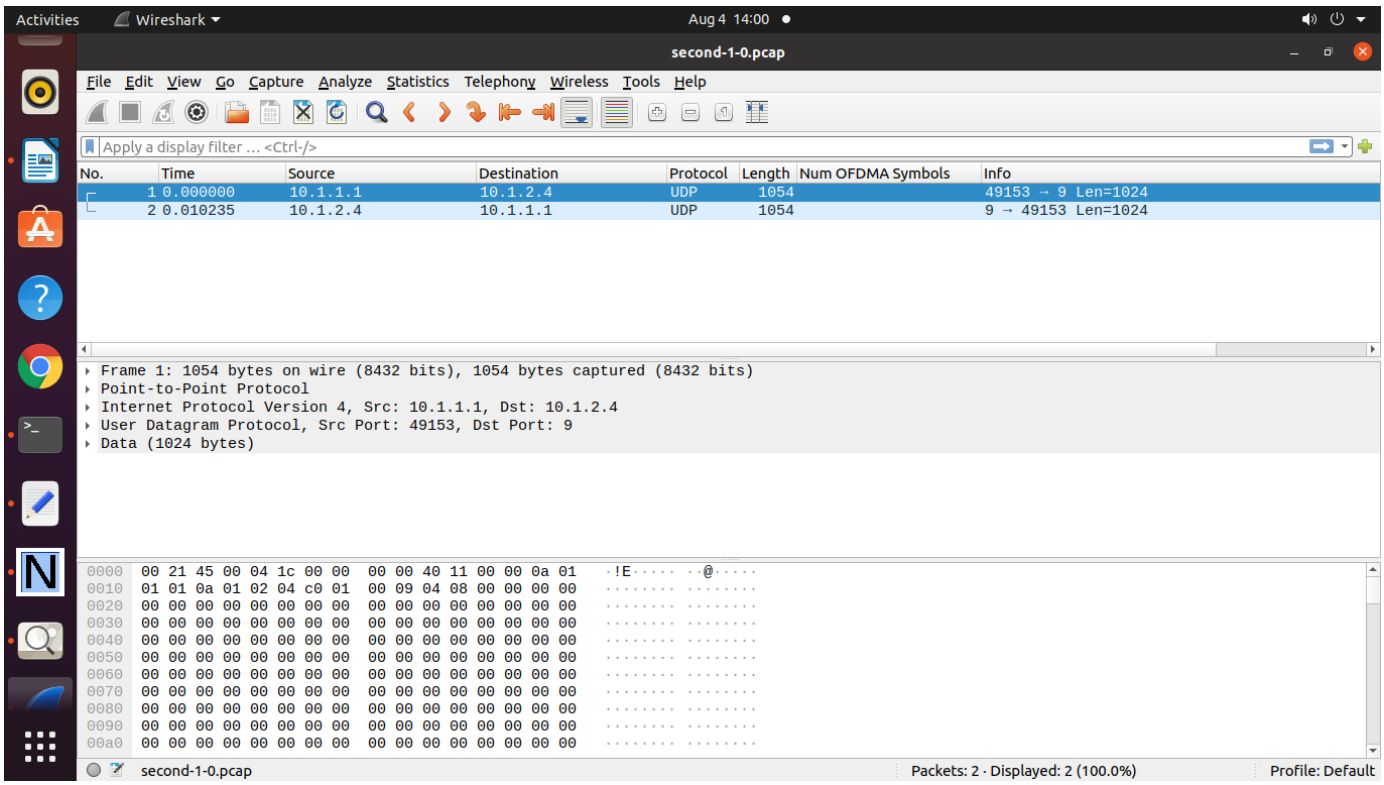
```
  Simulator::Destroy ();
  return 0;
}
```

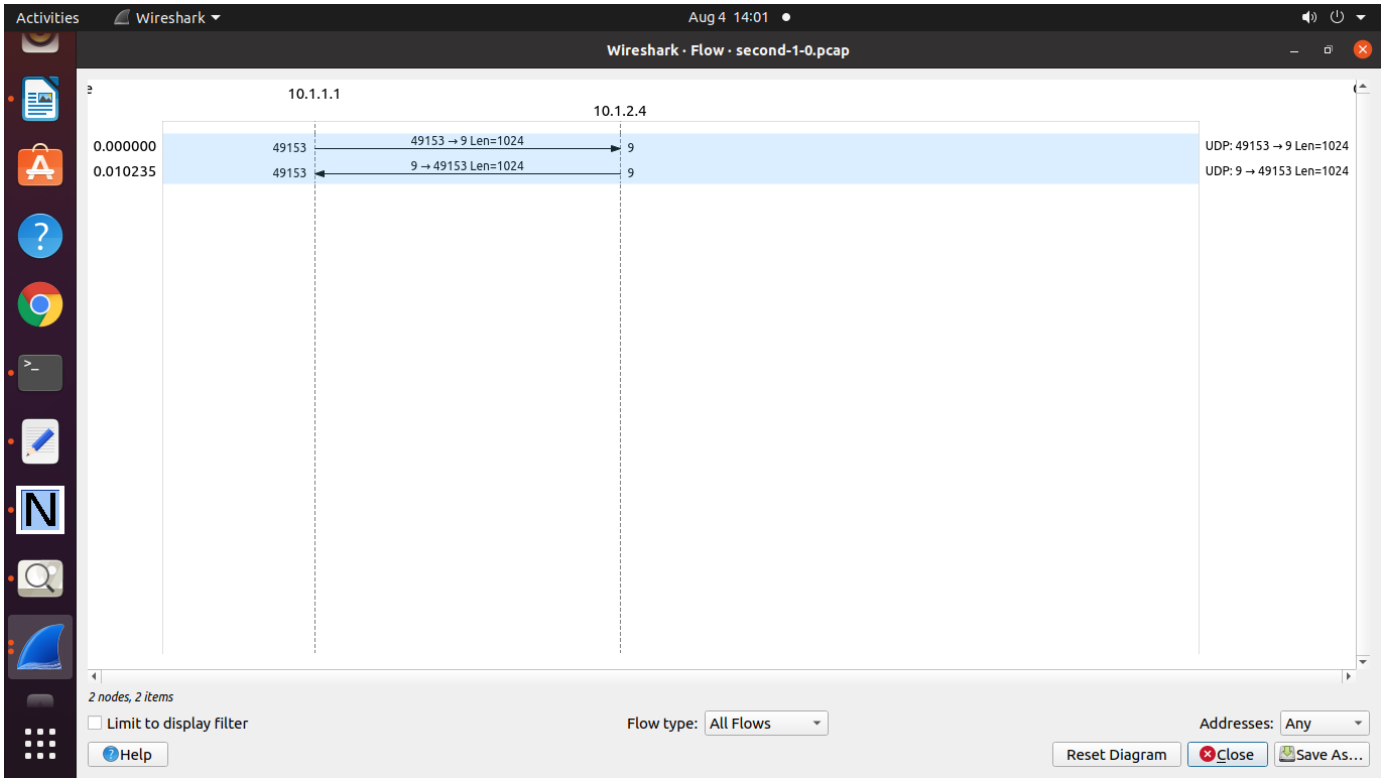
Output :






Activities
NetAnim
Aug 4 13:50
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
31 GB Vo...
se
Name
Location
Size
Type
Modified
LICENSE
18.0 kB
Text
7 Oct 2020
RELEASE_NOTES
171.8 kB
Text
7 Oct 2020
second.xml
14.4 kB
Markup
13:41
second-0-0.pcap
2.2 kB
Packet Capture (PCAP)
13:41
second-1-0.pcap
2.2 kB
Packet Capture (PCAP)
13:41
second-2-0.pcap
2.5 kB
Packet Capture (PCAP)
13:41
seq.txt.pcap
2.2 kB
Packet Capture (PCAP)
23 Jun
Activities
NetAnim
Aug 4 13:52
NetAnim
Animator
Stats
Packets
Activities
Terminal
Aug 4 13:54
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-...
csma-ping-0-0.pcap
csma-ping-1-0.pcap
csma-ping-2-0.pcap
csma-ping-3-0.pcap
first-0-0.pcap
first-1-0.pcap
gridtopology-0-0.pcap
gridtopology-0-1.pcap
gridtopology-10-0.pcap
gridtopology-10-1.pcap
gridtopology-10-2.pcap
gridtopology-1-0.pcap
gridtopology-11-0.pcap
gridtopology-11-1.pcap
gridtopology-11-2.pcap
gridtopology-11-3.pcap
gridtopology-1-1.pcap
gridtopology-12-0.pcap
gridtopology-12-1.pcap
gridtopology-12-2.pcap
gridtopology-12-3.pcap
gridtopology-1-2.pcap
gridtopology-13-0.pcap
Activities
Terminal
Aug 4 14:00
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ wireshark first-0-0.pcap
^C
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ wireshark second-1-0.pcap
ogy-6-0.pcap
ogy-7-0.pcap
ogy-8-0.pcap
.pcap
.pcap
.pcap
..pcap
1.pcap
1.pcap
0-1.pcap
1-1.pcap
2-1.pcap
3-1.pcap
.pcap
.pcap
.pcap
20-1.pcap
21-0.pcap


Activities
Wireshark
Aug 4 14:00
second-1-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Num OFDMA Symbols Info
1 0.000000 10.1.1.1 10.1.2.4 UDP 1054 49153 → 9 Len=1024
2 0.010235 10.1.2.4 10.1.1.1 UDP 1054 9 → 49153 Len=1024
Frame 1: 1054 bytes on wire (8432 bits), 1054 bytes captured (8432 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.1.1, Dst: 10.1.2.4
User Datagram Protocol, Src Port: 49153, Dst Port: 9
Data (1024 bytes)
0000 00 21 45 00 04 1c 00 00 00 00 40 11 00 00 0a 01
0010 01 01 0a 01 02 04 c0 01 00 09 04 08 00 00 00 00
second-1-0.pcap
Packets: 2 · Displayed: 2 (100.0%)
Profile: Default


Activities
Wireshark
Aug 4 14:01
Wireshark · Flow · second-1-0.pcap
10.1.1.1
10.1.2.4
0.000000 49153 49153 → 9 Len=1024 9 UDP: 49153 → 9 Len=1024
0.010235 49153 9 → 49153 Len=1024 9 UDP: 9 → 49153 Len=1024
2 nodes, 2 items
Limit to display filter
Flow type: All Flows
Addresses: Any
Help
Reset Diagram
Close
Save As...

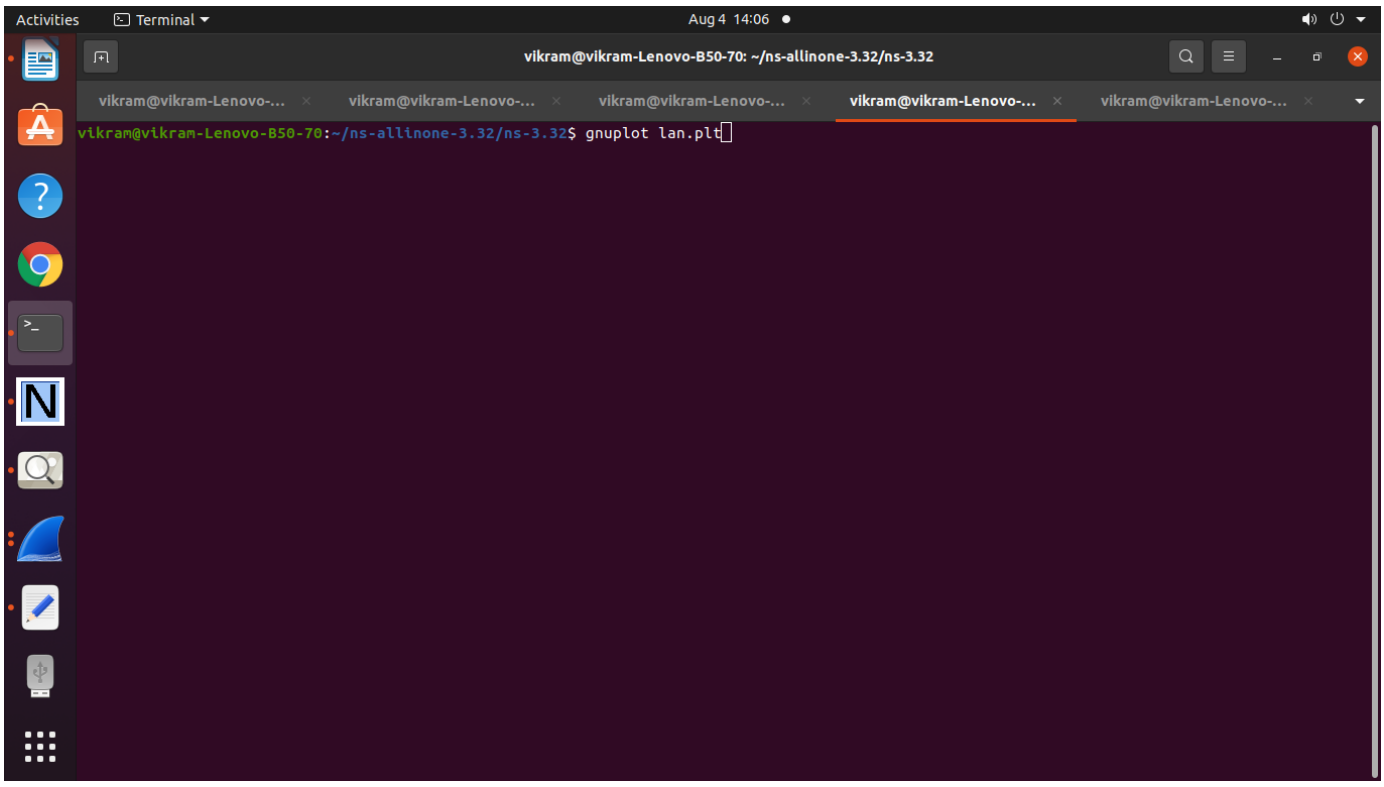
To draw a Graph

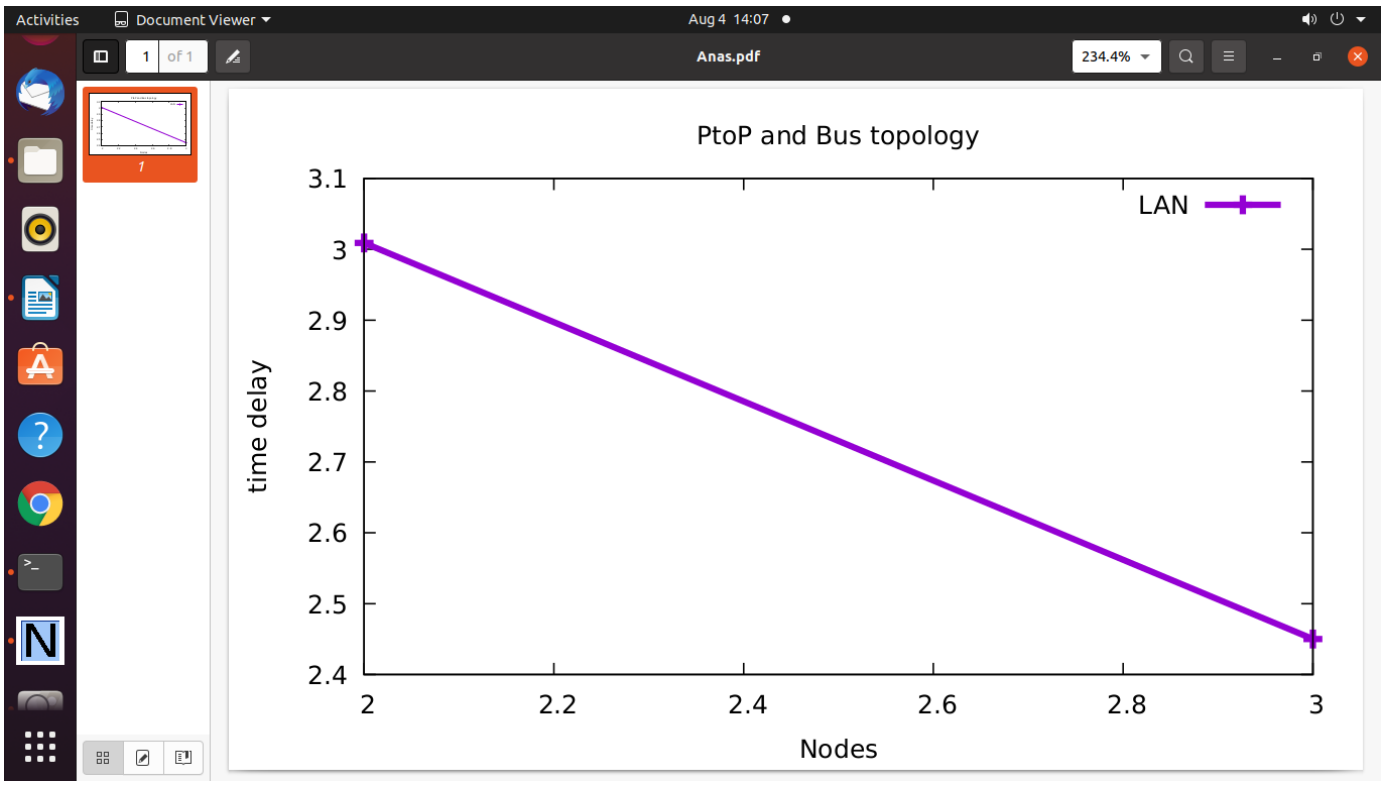
lan.txt

2 123 345 2.0067 3.009
3 123 45 1.09 2.45
4
5
---

lan.plt

```
set terminal pdf
set output "Anas.pdf"
set title "PtoP and Bus topology"
set xlabel "Nodes"
set ylabel "time delay"
plot "aaa.txt" using 1: 5 with linespoint title "LAN" lw 4
```




Activities
Document Viewer
Aug 4 14:07
1 of 1
Anas.pdf
234.4%
PtoP and Bus topology
LAN
time delay
Nodes
3.1
3
2.9
2.8
2.7
2.6
2.5
2.4
2
2.2
2.4
2.6
2.8
3

==========

Practical 9: Program to Simulate Hybrid LAN with Wi-Fi

```
/* -*- 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/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"

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



// Default Network Topology
//
//   Wifi 10.1.3.0
//                 AP
//  *    *    *    *
//  |    |    |    |    10.1.1.0
// n5   n6   n7   n0 -------------- n1   n2   n3   n4
//                   point-to-point  |    |    |    |
//                                   ================
//                                     LAN 10.1.2.0
```

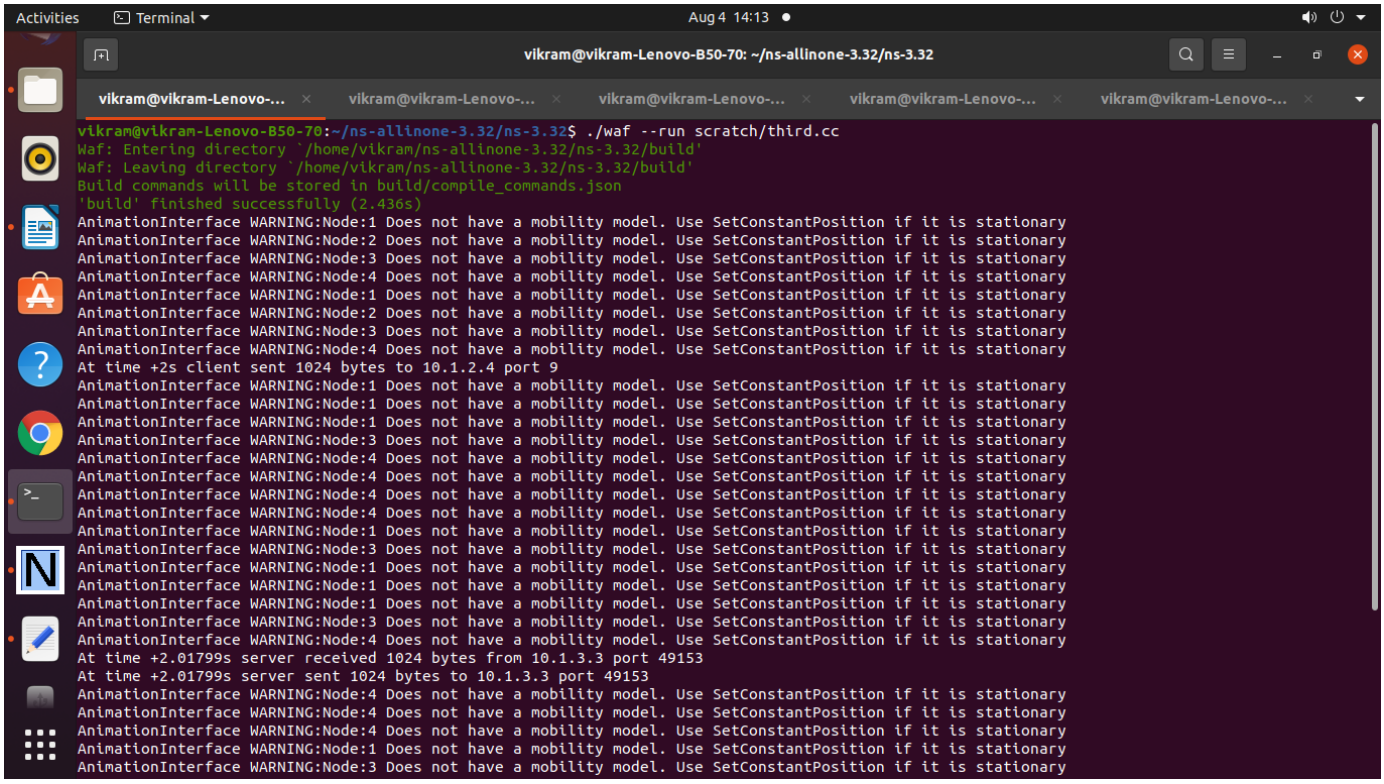
```
using namespace ns3;

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 \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  cmd.AddValue ("tracing", "Enable pcap tracing", tracing);

  cmd.Parse (argc,argv);

  // The underlying restriction of 18 is due to the grid position
  // allocator's configuration; the grid layout will exceed the
  // bounding box if more than 18 nodes are provided.
  if (nWifi > 18)
    {
      std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the
bounding box" << std::endl;
      return 1;
    }

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

  NodeContainer p2pNodes;
  p2pNodes.Create (2);

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

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

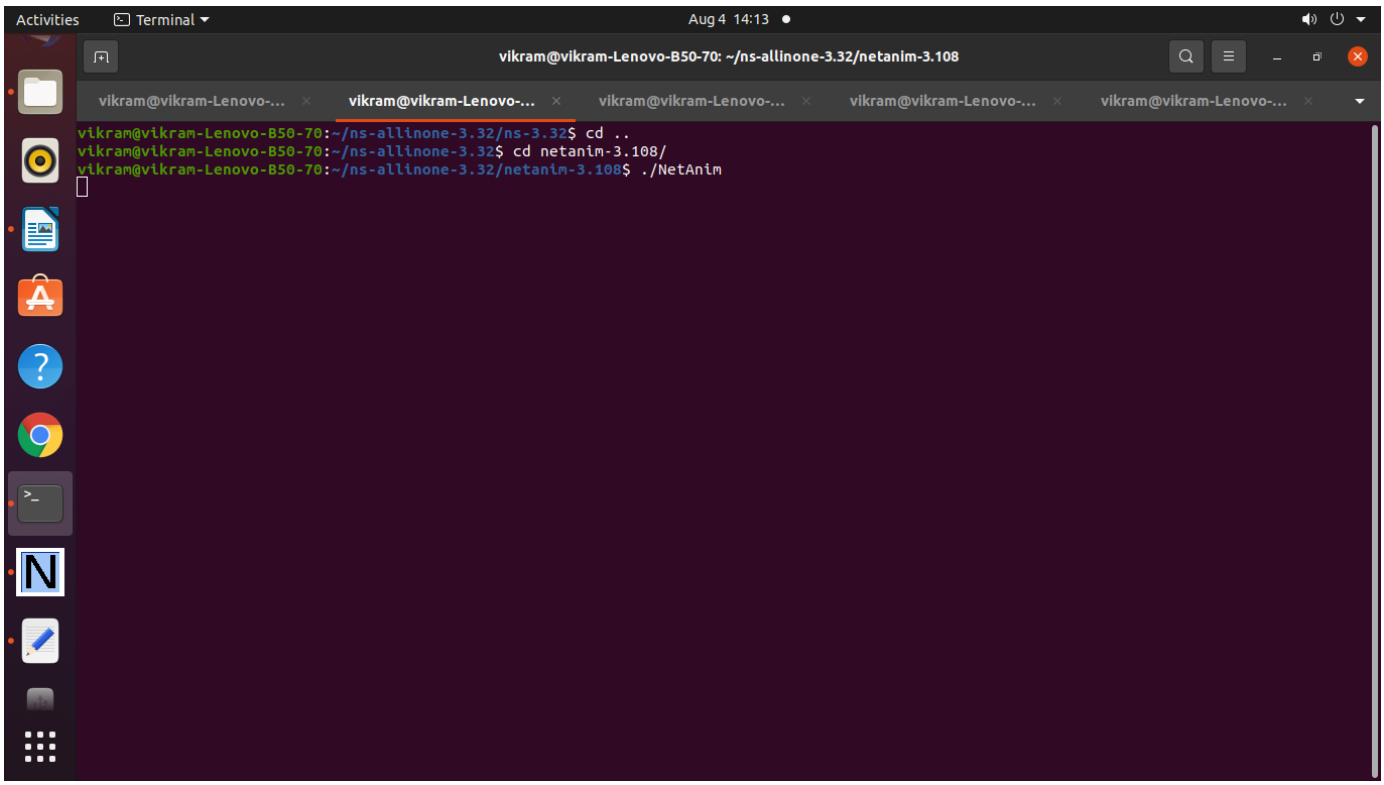
```
NodeContainer csmaNodes;
csmaNodes.Add (p2pNodes.Get (1));
csmaNodes.Create (nCsma);

CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);

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");
mac.SetType ("ns3::StaWifiMac",
             "Ssid", SsidValue (ssid),
             "ActiveProbing", BooleanValue (false));

NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);

mac.SetType ("ns3::ApWifiMac",
             "Ssid", SsidValue (ssid));

NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);

MobilityHelper mobility;

mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                               "MinX", DoubleValue (0.0),
                               "MinY", DoubleValue (0.0),
                               "DeltaX", DoubleValue (5.0),
                               "DeltaY", DoubleValue (10.0),
```

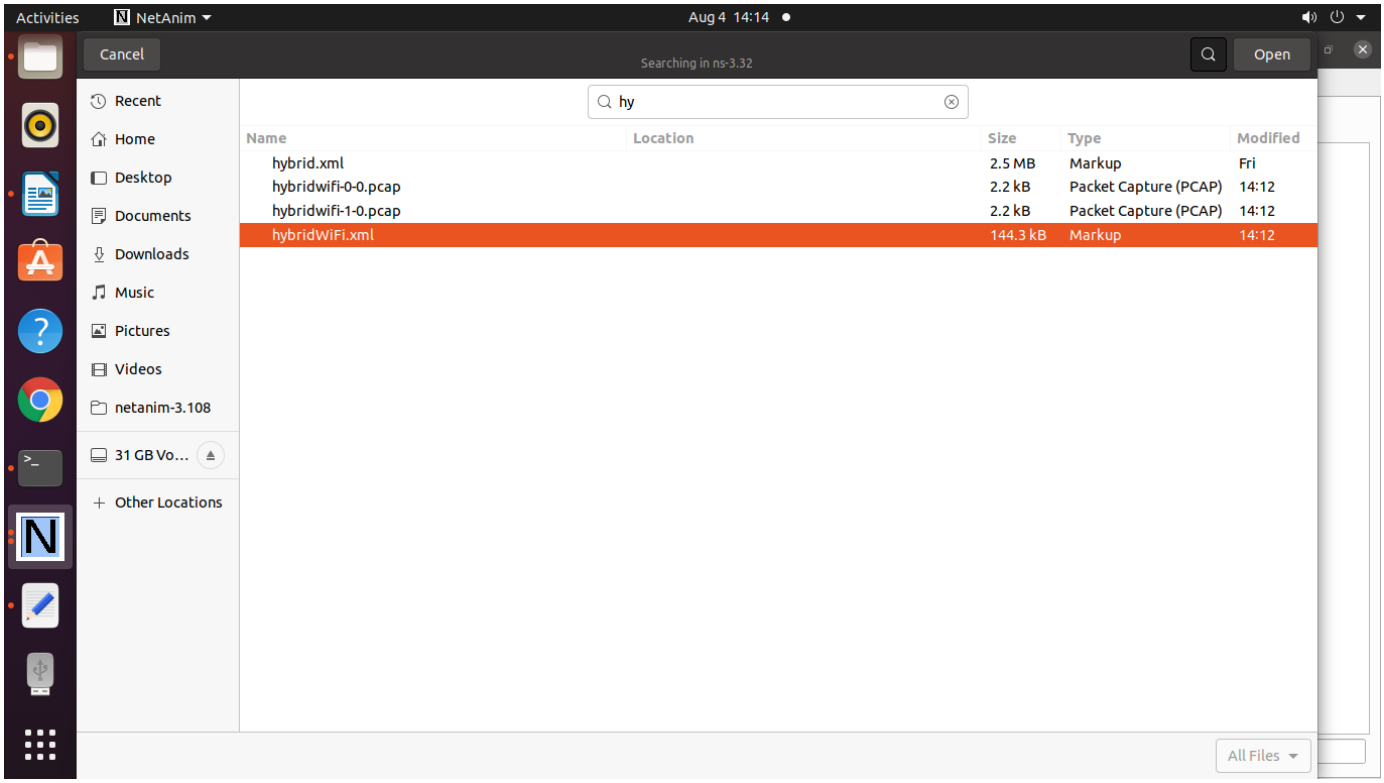
```
                    "GridWidth", UintegerValue (3),
                    "LayoutType", StringValue ("RowFirst"));

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

mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);

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;
p2pInterfaces = address.Assign (p2pDevices);

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

address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);

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 (wifiStaNodes.Get (nWifi - 1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
```

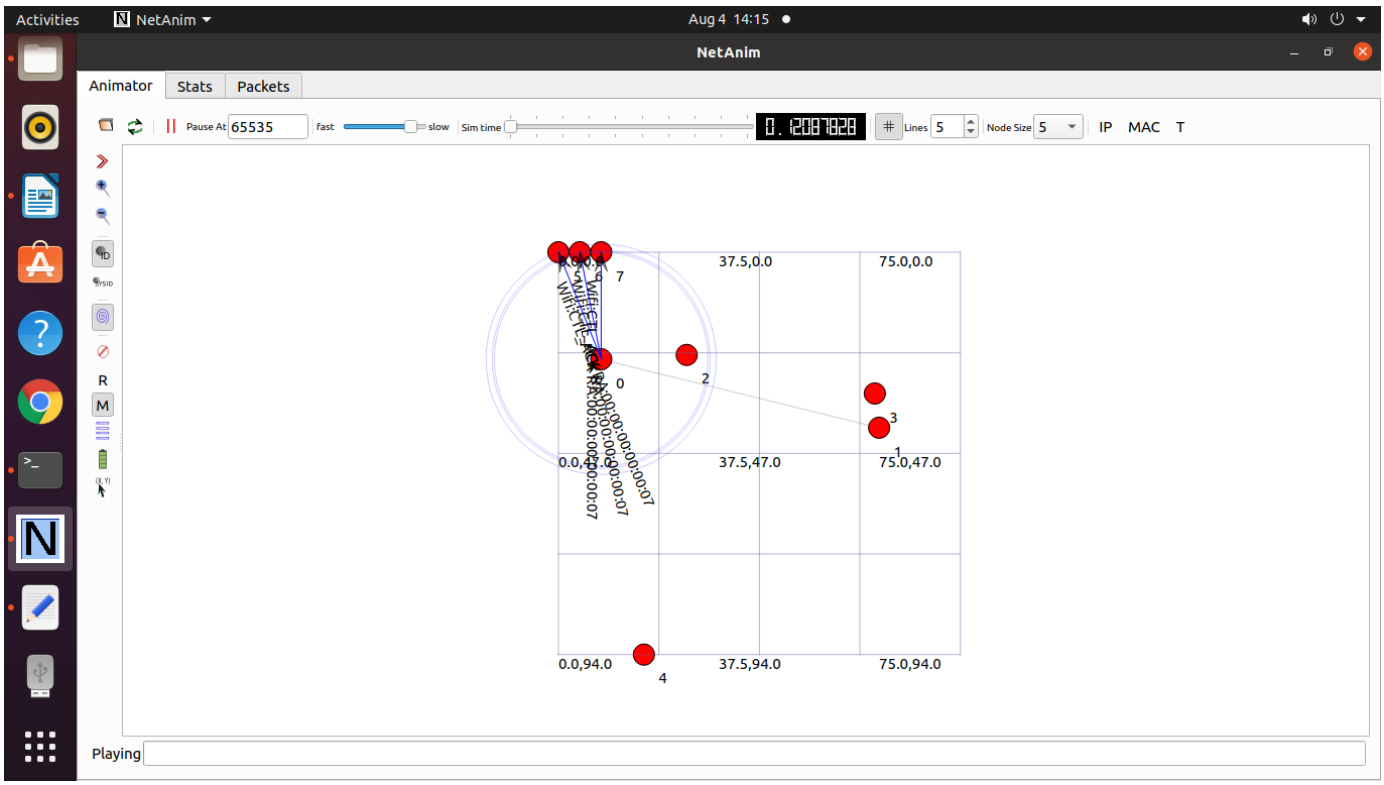
```
  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
  AnimationInterface anim("hybridWiFi.xml");
AnimationInterface::SetConstantPosition (p2pNodes.Get(0), 10, 25);
AnimationInterface ::SetConstantPosition(csmaNodes.Get(1), 40,25);
anim.EnablePacketMetadata(true);
pointToPoint.EnablePcapAll("hybridwifi");

  Simulator::Stop (Seconds (10.0));

  if (tracing == true)
    {
      pointToPoint.EnablePcapAll ("third");
      phy.EnablePcap ("third", apDevices.Get (0));
      csma.EnablePcap ("third", csmaDevices.Get (0), true);
    }

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

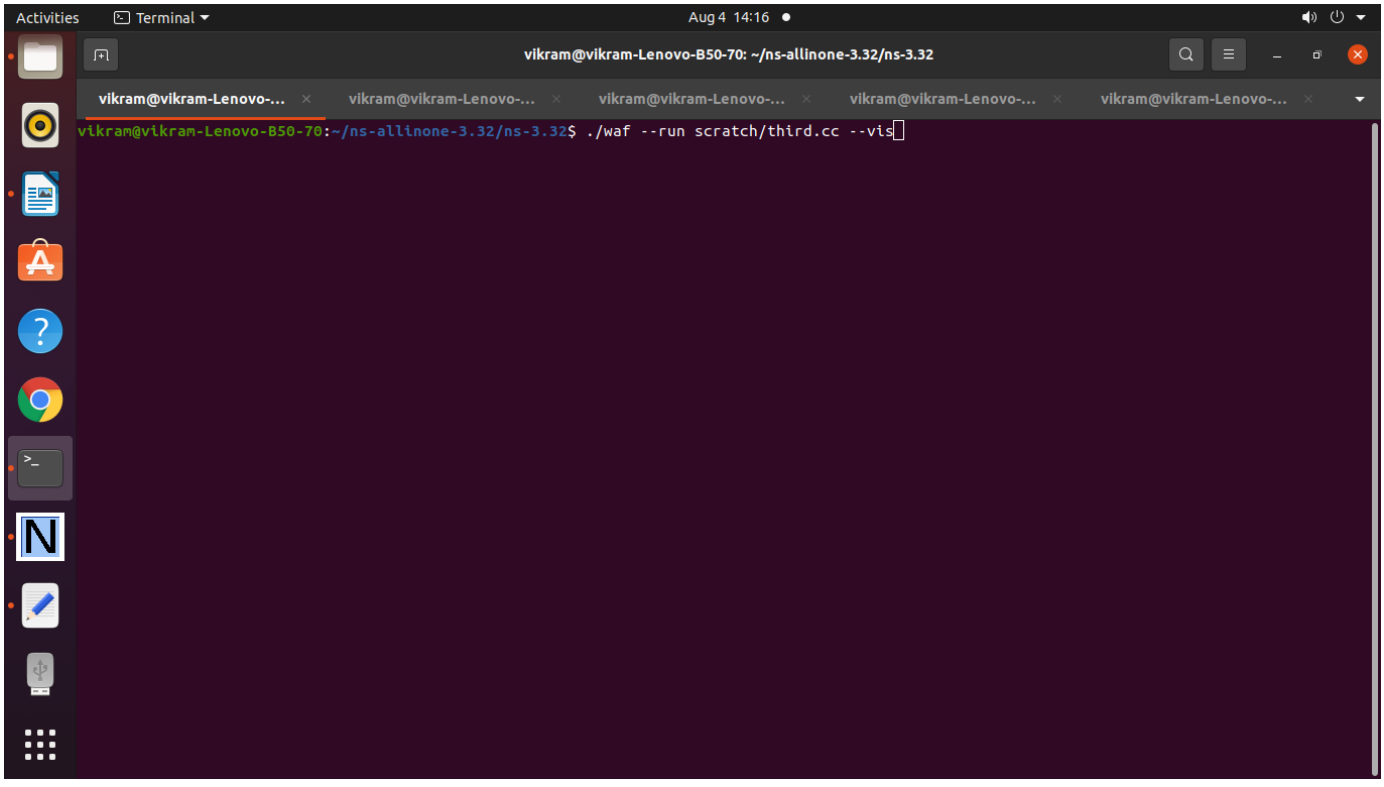
Output :

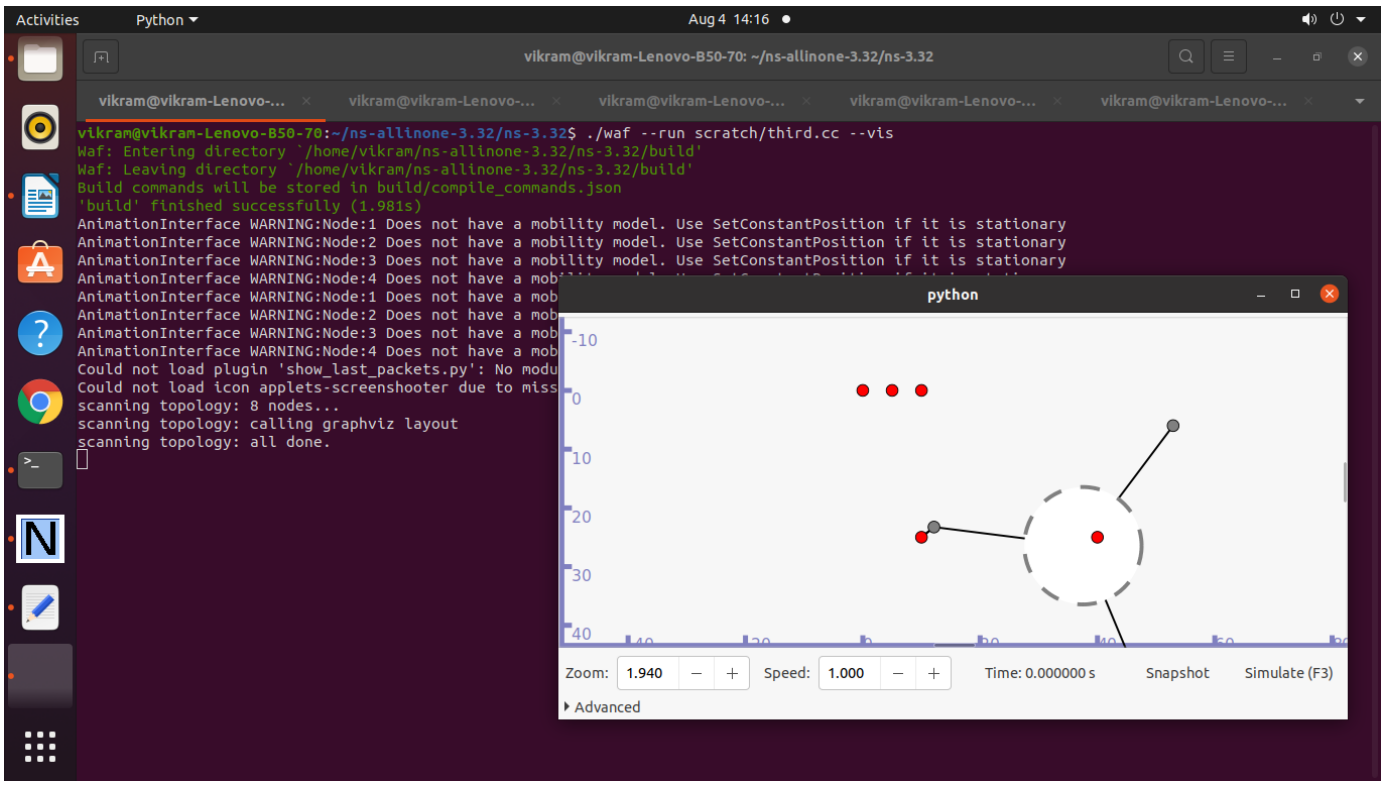


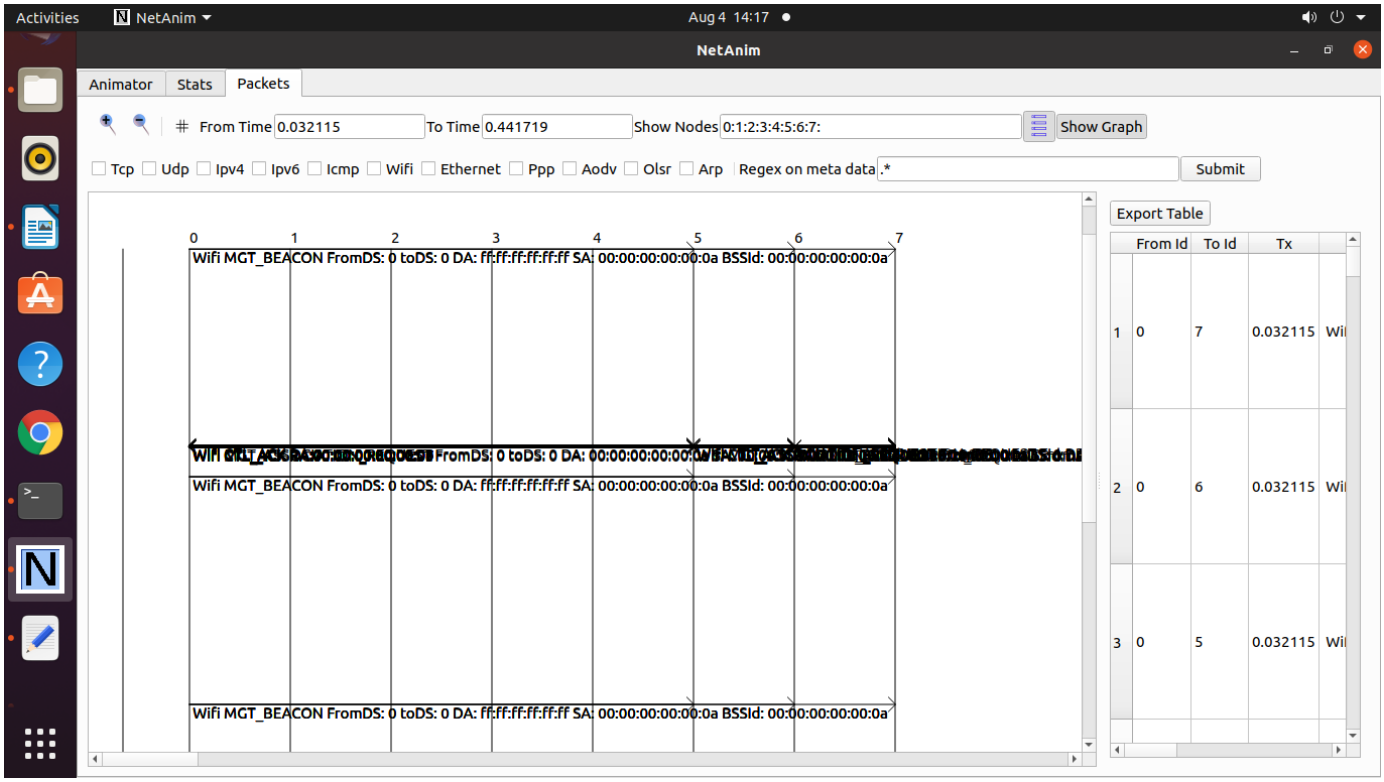

Activities
Terminal
Aug 4 14:13
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/netanim-3.108
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ cd ..
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32$ cd netanim-3.108/
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/netanim-3.108$ ./NetAnim

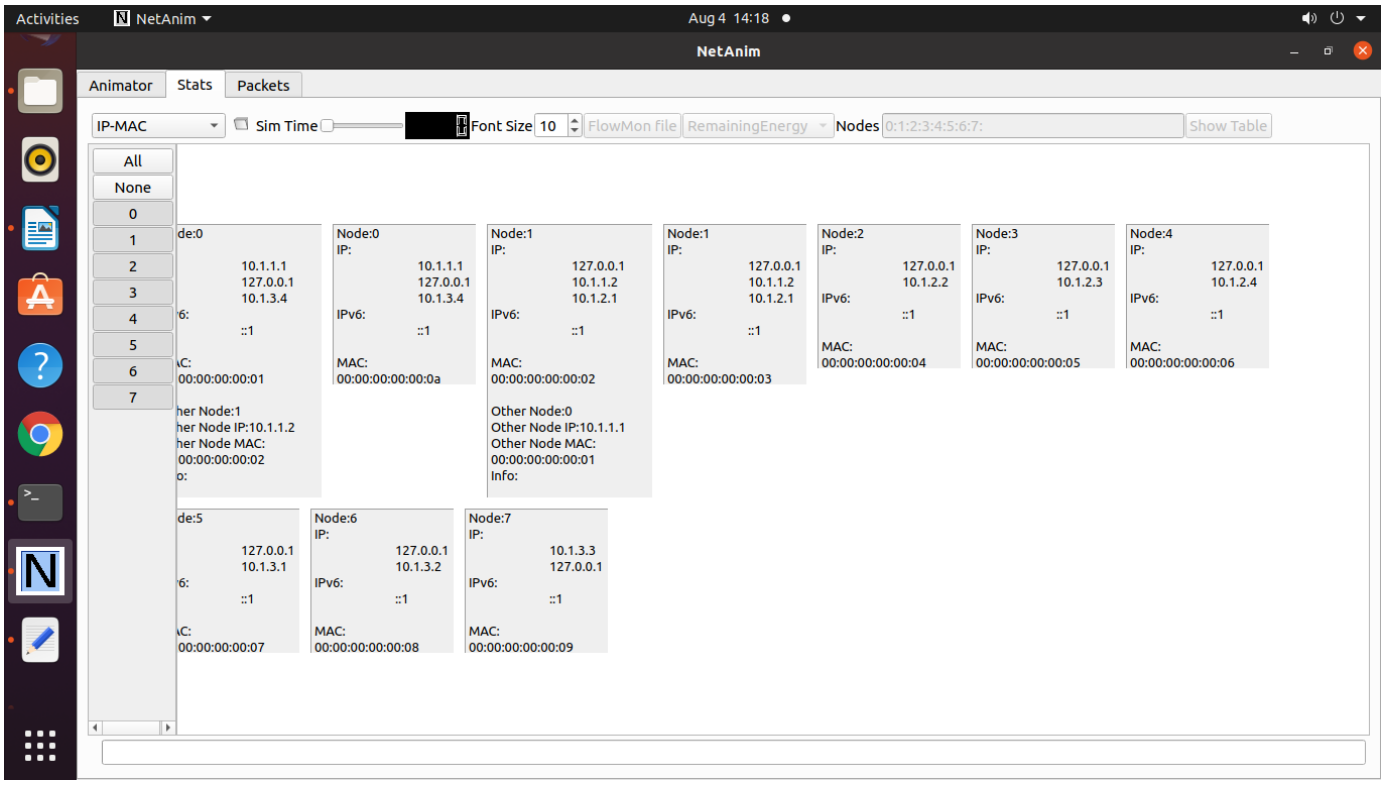

Activities
NetAnim
Aug 4 14:14
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
31 GB Vo...
Other Locations
hy
Name
Location
Size
Type
Modified
hybrid.xml
2.5 MB
Markup
Fri
hybridwifi-0-0.pcap
2.2 kB
Packet Capture (PCAP)
14:12
hybridwifi-1-0.pcap
2.2 kB
Packet Capture (PCAP)
14:12
hybridWiFi.xml
144.3 kB
Markup
14:12
All Files

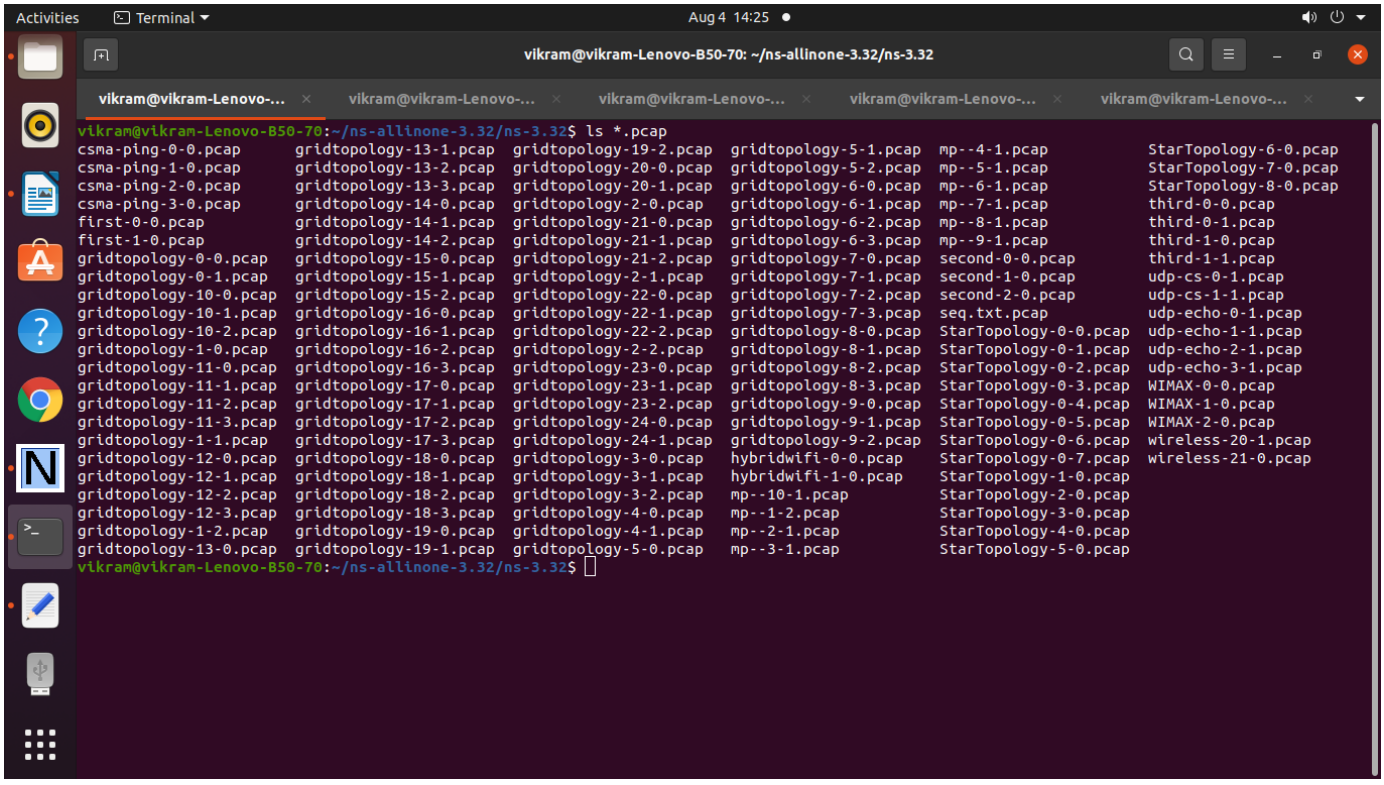

Activities
NetAnim
Aug 4 14:15
NetAnim
Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
Lines 5
Node Size 5
IP
MAC
T
37.5,0.0
75.0,0.0
0.0,47.0
37.5,47.0
75.0,47.0
0.0,94.0
37.5,94.0
75.0,94.0
0
1
2
3
4
7
Playing

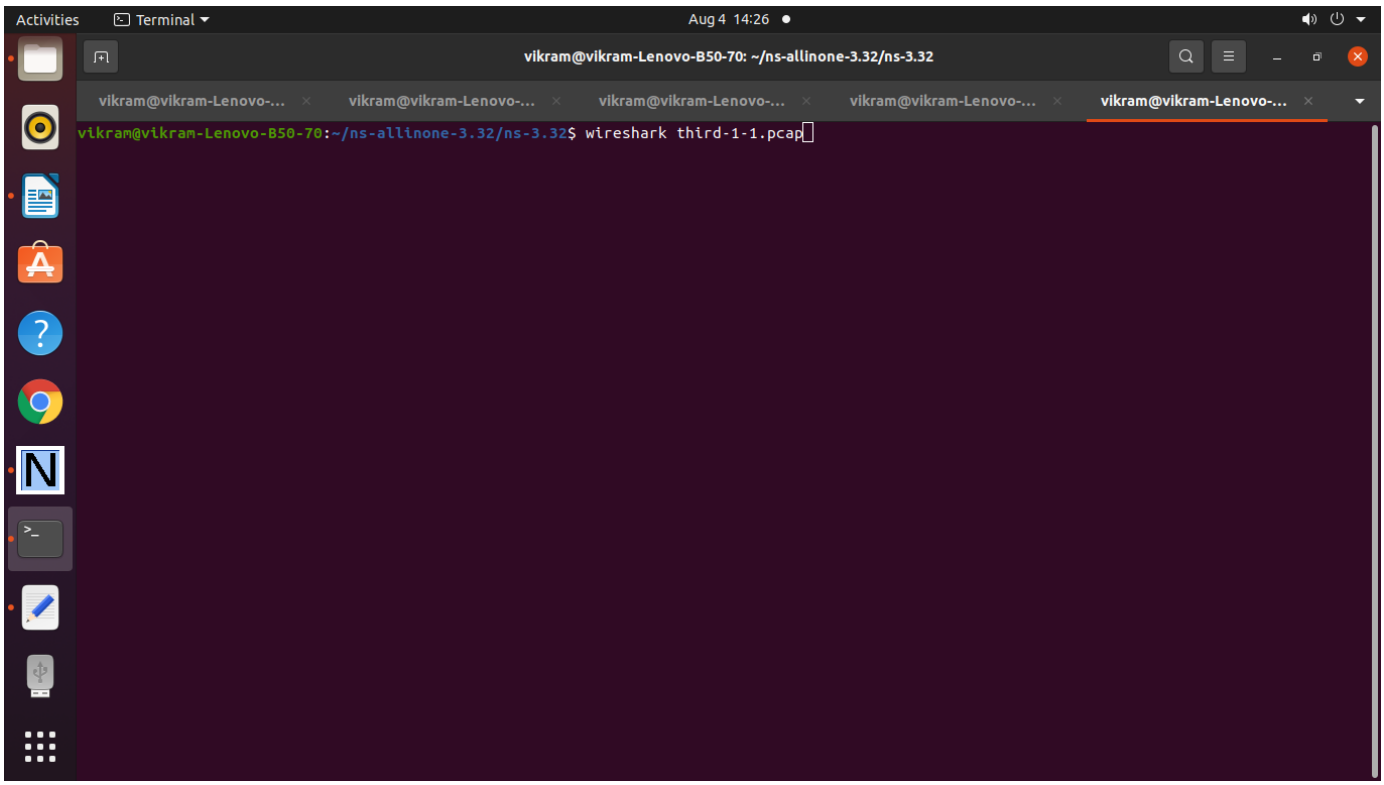

Activities
Terminal
Aug 4 14:16
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/third.cc --vis

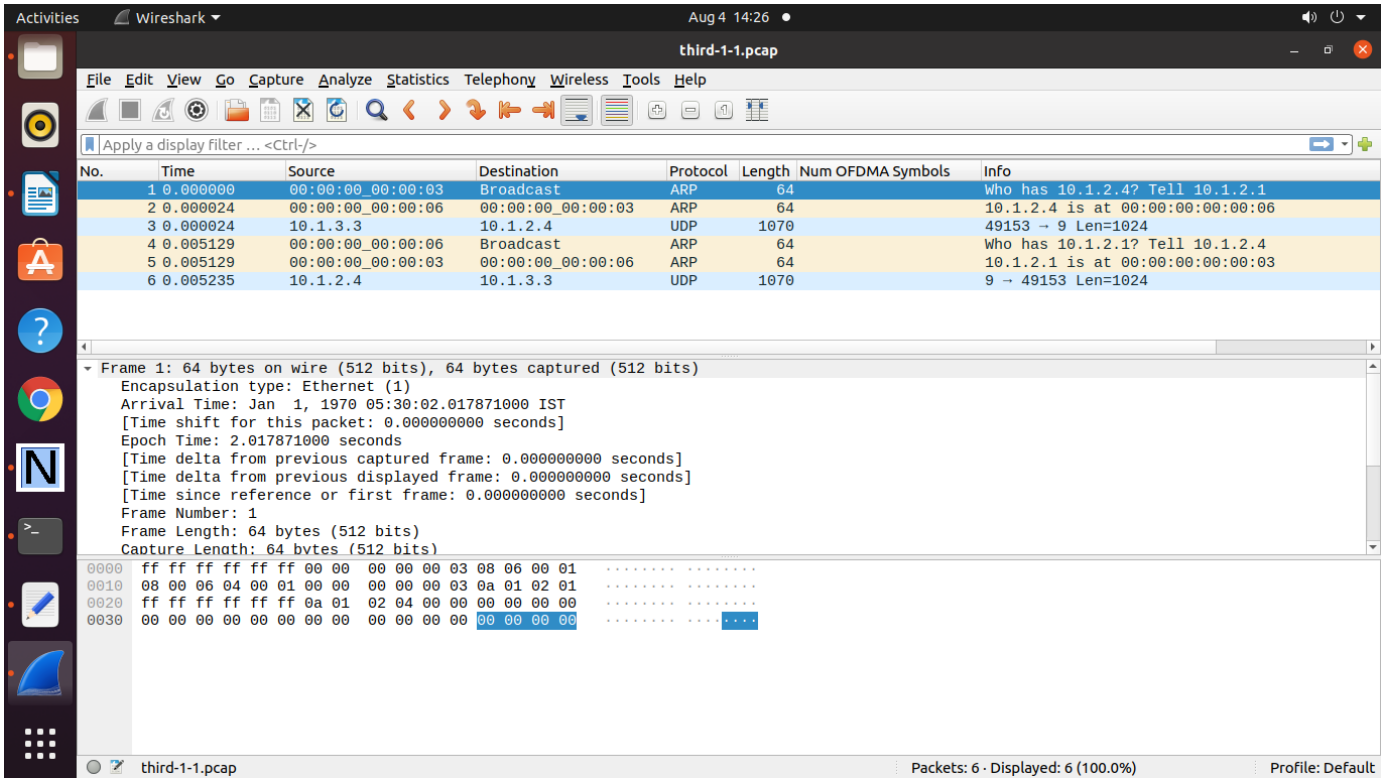

Activities
Python
Aug 4 14:16
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/third.cc --vis
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.981s)
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mob
AnimationInterface WARNING:Node:2 Does not have a mob
AnimationInterface WARNING:Node:3 Does not have a mob
AnimationInterface WARNING:Node:4 Does not have a mob
Could not load plugin 'show_last_packets.py': No modu
Could not load icon applets-screenshooter due to miss
scanning topology: 8 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
python
-10
0
10
20
30
40
Zoom: 1.940
Speed: 1.000
Time: 0.000000 s
Snapshot
Simulate (F3)
Advanced

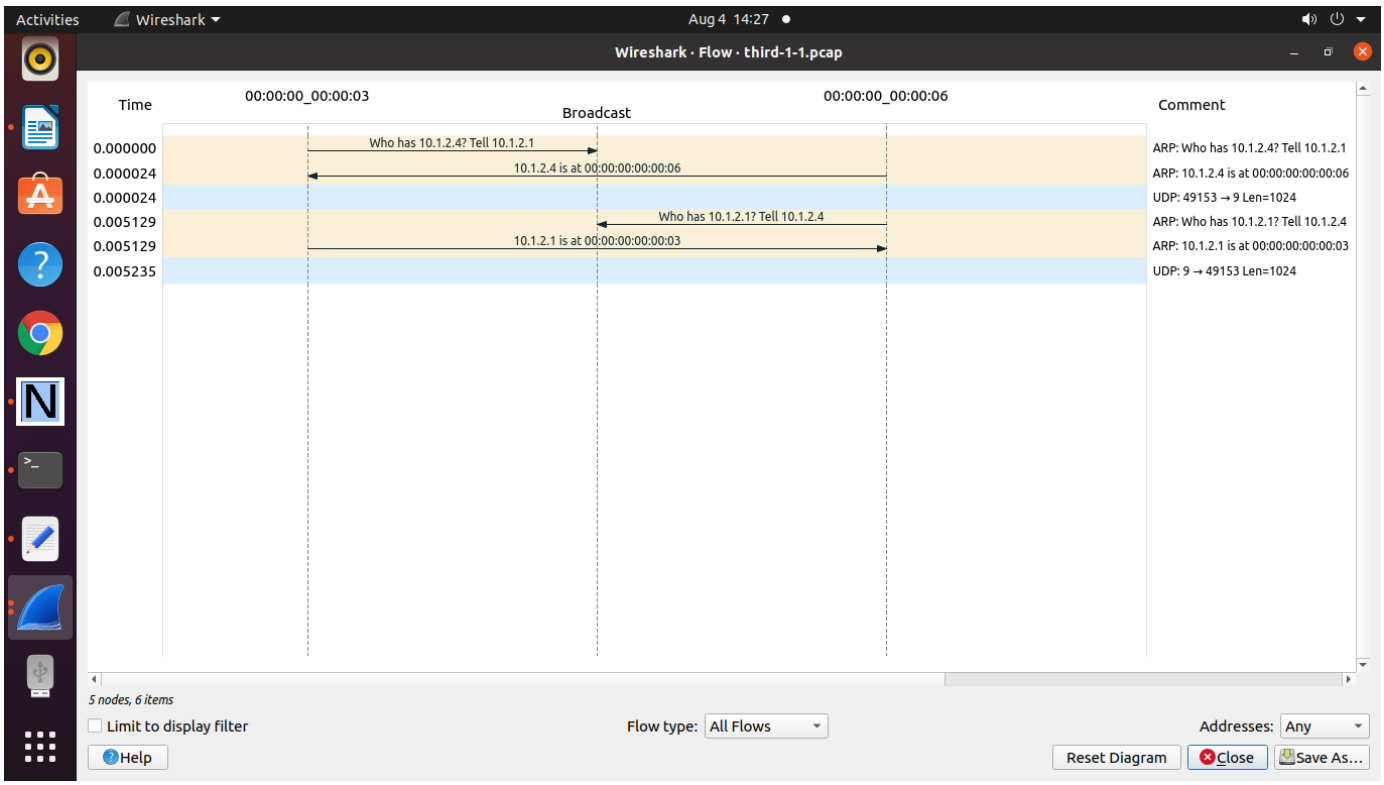

Activities
NetAnim
Aug 4 14:17
NetAnim
Animator
Stats
Packets
From Time 0.032115
To Time 0.441719
Show Nodes 0:1:2:3:4:5:6:7:
Show Graph
Tcp
Udp
Ipv4
Ipv6
Icmp
Wifi
Ethernet
Ppp
Aodv
Olsr
Arp
Regex on meta data .*
Submit
0
1
2
3
4
5
6
7
Wifi MGT_BEACON FromDS: 0 toDS: 0 DA: ff:ff:ff:ff:ff:ff SA: 00:00:00:00:00:0a BSSId: 00:00:00:00:00:0a
Wifi MGT_BEACON FromDS: 0 toDS: 0 DA: ff:ff:ff:ff:ff:ff SA: 00:00:00:00:00:0a BSSId: 00:00:00:00:00:0a
Wifi MGT_BEACON FromDS: 0 toDS: 0 DA: ff:ff:ff:ff:ff:ff SA: 00:00:00:00:00:0a BSSId: 00:00:00:00:00:0a
Export Table
From Id
To Id
Tx
1 0 7 0.032115 Wi
2 0 6 0.032115 Wi
3 0 5 0.032115 Wi


Activities
NetAnim
Aug 4 14:18
NetAnim
Animator
Stats
Packets
IP-MAC
Sim Time
Font Size
10
FlowMon file
RemainingEnergy
Nodes
0:1:2:3:4:5:6:7:
Show Table
All
None
0
1
2
3
4
5
6
7
Node:0
IP:
10.1.1.1
127.0.0.1
10.1.3.4
IPv6:
::1
MAC:
00:00:00:00:00:0a
Node:1
IP:
127.0.0.1
10.1.1.2
10.1.2.1
IPv6:
::1
MAC:
00:00:00:00:00:02
Other Node:0
Other Node IP:10.1.1.1
Other Node MAC:
00:00:00:00:00:01
Info:
Node:1
IP:
127.0.0.1
10.1.1.2
10.1.2.1
IPv6:
::1
MAC:
00:00:00:00:00:03
Node:2
IP:
127.0.0.1
10.1.2.2
IPv6:
::1
MAC:
00:00:00:00:00:04
Node:3
IP:
127.0.0.1
10.1.2.3
IPv6:
::1
MAC:
00:00:00:00:00:05
Node:4
IP:
127.0.0.1
10.1.2.4
IPv6:
::1
MAC:
00:00:00:00:00:06
Node:6
IP:
127.0.0.1
10.1.3.2
IPv6:
::1
MAC:
00:00:00:00:00:08
Node:7
IP:
10.1.3.3
127.0.0.1
IPv6:
::1
MAC:
00:00:00:00:00:09


Activities
Terminal
Aug 4 14:25
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ls *.pcap
csma-ping-0-0.pcap gridtopology-13-1.pcap gridtopology-19-2.pcap gridtopology-5-1.pcap mp--4-1.pcap StarTopology-6-0.pcap
csma-ping-1-0.pcap gridtopology-13-2.pcap gridtopology-20-0.pcap gridtopology-5-2.pcap mp--5-1.pcap StarTopology-7-0.pcap
csma-ping-2-0.pcap gridtopology-13-3.pcap gridtopology-20-1.pcap gridtopology-6-0.pcap mp--6-1.pcap StarTopology-8-0.pcap
csma-ping-3-0.pcap gridtopology-14-0.pcap gridtopology-2-0.pcap gridtopology-6-1.pcap mp--7-1.pcap third-0-0.pcap
first-0-0.pcap gridtopology-14-1.pcap gridtopology-21-0.pcap gridtopology-6-2.pcap mp--8-1.pcap third-0-1.pcap
first-1-0.pcap gridtopology-14-2.pcap gridtopology-21-1.pcap gridtopology-6-3.pcap mp--9-1.pcap third-1-0.pcap
gridtopology-0-0.pcap gridtopology-15-0.pcap gridtopology-21-2.pcap gridtopology-7-0.pcap second-0-0.pcap third-1-1.pcap
gridtopology-0-1.pcap gridtopology-15-1.pcap gridtopology-2-1.pcap gridtopology-7-1.pcap second-1-0.pcap udp-cs-0-1.pcap
gridtopology-10-0.pcap gridtopology-15-2.pcap gridtopology-22-0.pcap gridtopology-7-2.pcap second-2-0.pcap udp-cs-1-1.pcap
gridtopology-10-1.pcap gridtopology-16-0.pcap gridtopology-22-1.pcap gridtopology-7-3.pcap seq.txt.pcap udp-echo-0-1.pcap
gridtopology-10-2.pcap gridtopology-16-1.pcap gridtopology-22-2.pcap gridtopology-8-0.pcap StarTopology-0-0.pcap udp-echo-1-1.pcap
gridtopology-1-0.pcap gridtopology-16-2.pcap gridtopology-2-2.pcap gridtopology-8-1.pcap StarTopology-0-1.pcap udp-echo-2-1.pcap
gridtopology-11-0.pcap gridtopology-16-3.pcap gridtopology-23-0.pcap gridtopology-8-2.pcap StarTopology-0-2.pcap udp-echo-3-1.pcap
gridtopology-11-1.pcap gridtopology-17-0.pcap gridtopology-23-1.pcap gridtopology-8-3.pcap StarTopology-0-3.pcap WIMAX-0-0.pcap
gridtopology-11-2.pcap gridtopology-17-1.pcap gridtopology-23-2.pcap gridtopology-9-0.pcap StarTopology-0-4.pcap WIMAX-1-0.pcap
gridtopology-11-3.pcap gridtopology-17-2.pcap gridtopology-24-0.pcap gridtopology-9-1.pcap StarTopology-0-5.pcap WIMAX-2-0.pcap
gridtopology-1-1.pcap gridtopology-17-3.pcap gridtopology-24-1.pcap gridtopology-9-2.pcap StarTopology-0-6.pcap wireless-20-1.pcap
gridtopology-12-0.pcap gridtopology-18-0.pcap gridtopology-3-0.pcap hybridwifi-0-0.pcap StarTopology-0-7.pcap wireless-21-0.pcap
gridtopology-12-1.pcap gridtopology-18-1.pcap gridtopology-3-1.pcap hybridwifi-1-0.pcap StarTopology-1-0.pcap
gridtopology-12-2.pcap gridtopology-18-2.pcap gridtopology-3-2.pcap mp--10-1.pcap StarTopology-2-0.pcap
gridtopology-12-3.pcap gridtopology-18-3.pcap gridtopology-4-0.pcap mp--1-2.pcap StarTopology-3-0.pcap
gridtopology-1-2.pcap gridtopology-19-0.pcap gridtopology-4-1.pcap mp--2-1.pcap StarTopology-4-0.pcap
gridtopology-13-0.pcap gridtopology-19-1.pcap gridtopology-5-0.pcap mp--3-1.pcap StarTopology-5-0.pcap
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$


Activities
Terminal
Aug 4 14:26
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ wireshark third-1-1.pcap

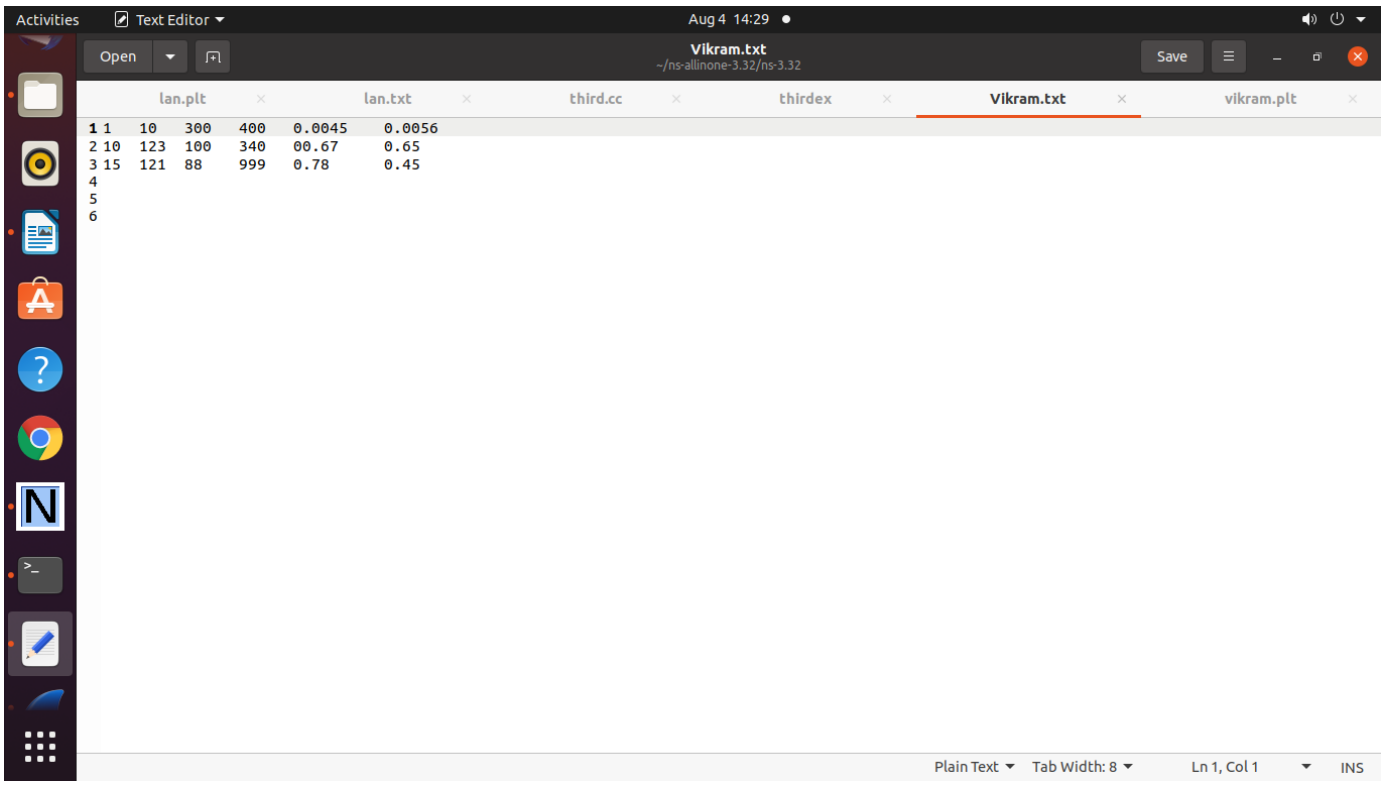

Activities
Wireshark
Aug 4 14:26
third-1-1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Num OFDMA Symbols Info
1 0.000000 00:00:00_00:00:03 Broadcast ARP 64 Who has 10.1.2.4? Tell 10.1.2.1
2 0.000024 00:00:00_00:00:06 00:00:00_00:00:03 ARP 64 10.1.2.4 is at 00:00:00:00:00:06
3 0.000024 10.1.3.3 10.1.2.4 UDP 1070 49153 → 9 Len=1024
4 0.005129 00:00:00_00:00:06 Broadcast ARP 64 Who has 10.1.2.1? Tell 10.1.2.4
5 0.005129 00:00:00_00:00:03 00:00:00_00:00:06 ARP 64 10.1.2.1 is at 00:00:00:00:00:03
6 0.005235 10.1.2.4 10.1.3.3 UDP 1070 9 → 49153 Len=1024
Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
Encapsulation type: Ethernet (1)
Arrival Time: Jan 1, 1970 05:30:02.017871000 IST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 2.017871000 seconds
[Time delta from previous captured frame: 0.000000000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
Frame Length: 64 bytes (512 bits)
Capture Length: 64 bytes (512 bits)
0000 ff ff ff ff ff ff 00 00 00 00 00 03 08 06 00 01
0010 08 00 06 04 00 01 00 00 00 00 00 03 0a 01 02 01
0020 ff ff ff ff ff ff 0a 01 02 04 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
third-1-1.pcap
Packets: 6 · Displayed: 6 (100.0%)
Profile: Default

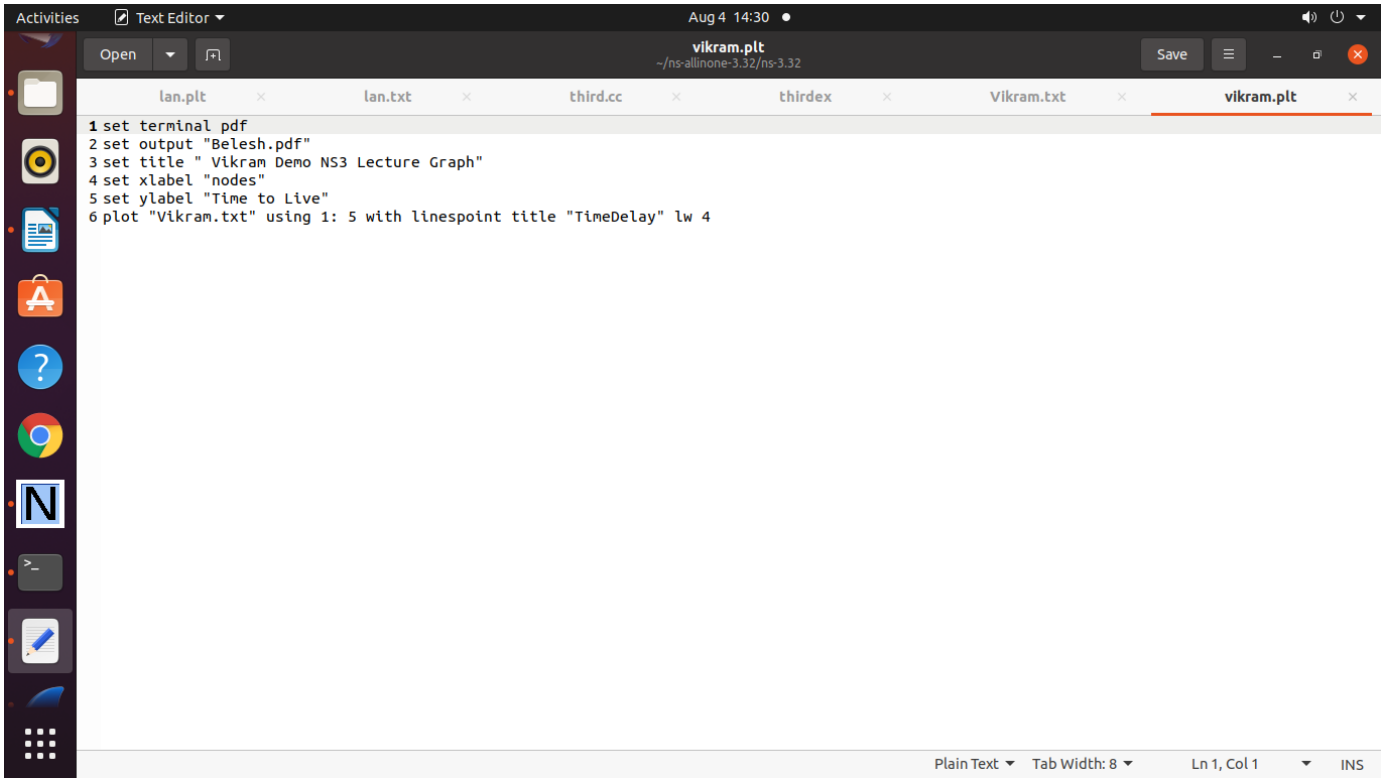


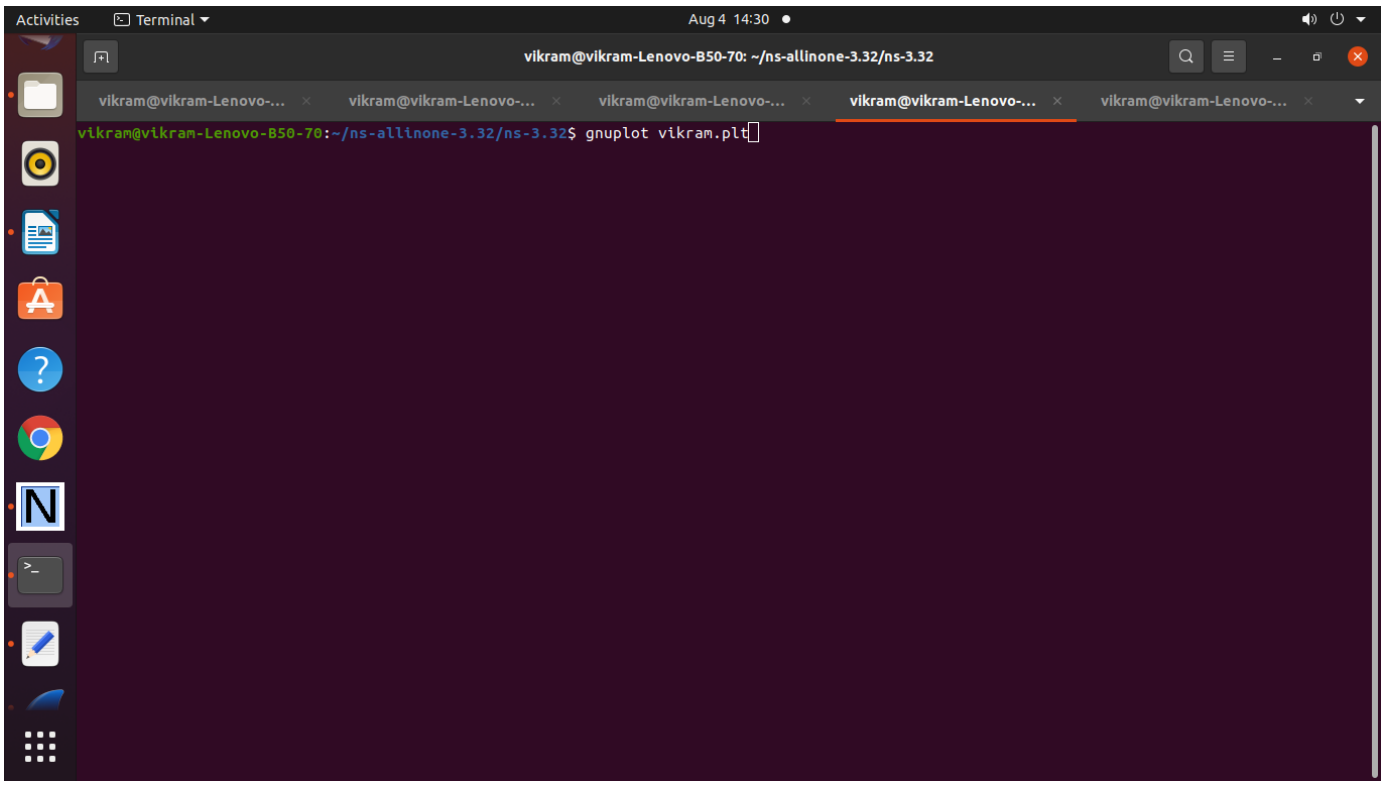
To Plot a Graph

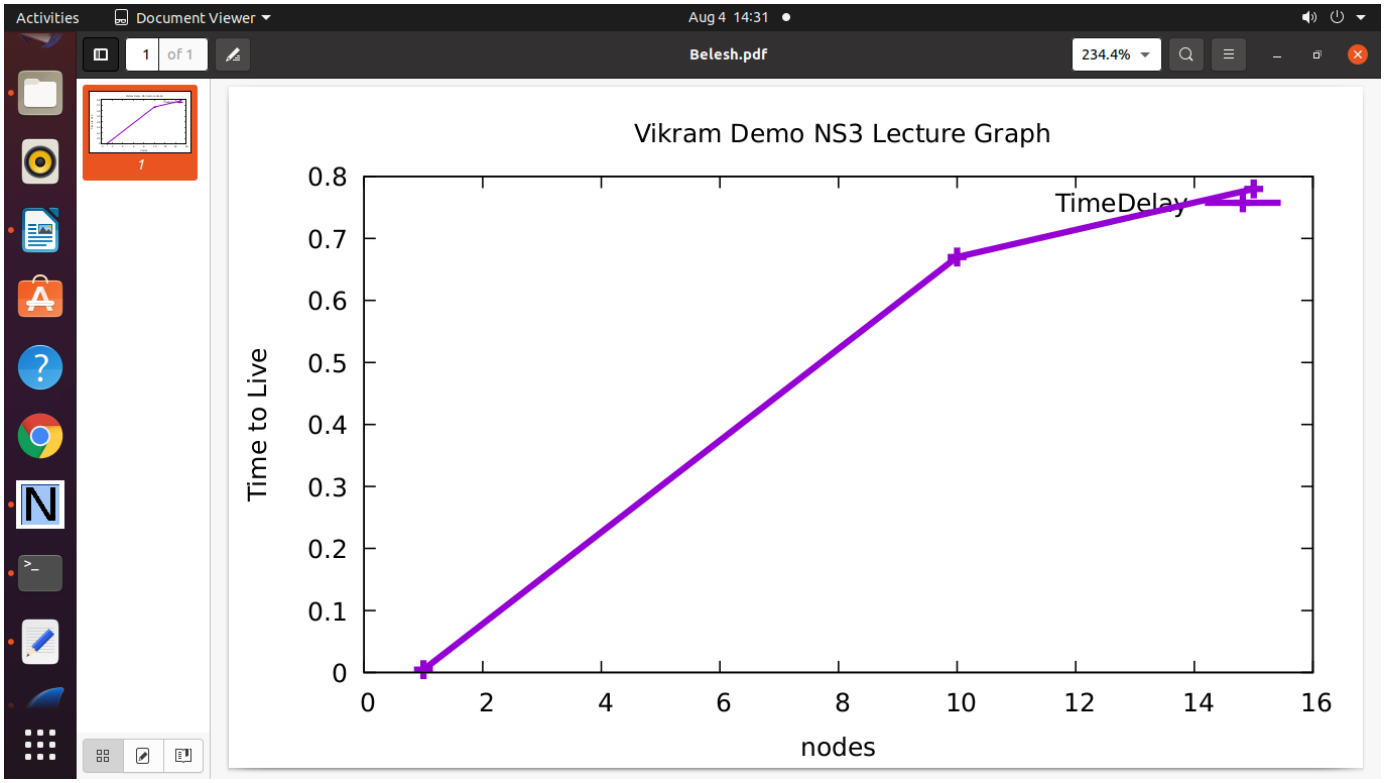
vikram.txt

1 10 300 400 0.0045 0.0056
10 123 100 340 00.67 0.65
15 121 88 999 0.78 0.45


Activities
Text Editor
Aug 4 14:29
Open
Vikram.txt
~/ns-allinone-3.32/ns-3.32
Save
lan.plt
lan.txt
third.cc
thirdex
Vikram.txt
vikram.plt
1 1 10 300 400 0.0045 0.0056
2 10 123 100 340 00.67 0.65
3 15 121 88 999 0.78 0.45
4
5
6
Plain Text
Tab Width: 8
Ln 1, Col 1
INS


Activities
Text Editor
Aug 4 14:30
Open
vikram.plt
~/ns-allinone-3.32/ns-3.32
Save
lan.plt
lan.txt
third.cc
thirdex
Vikram.txt
vikram.plt
1 set terminal pdf
2 set output "Belesh.pdf"
3 set title " Vikram Demo NS3 Lecture Graph"
4 set xlabel "nodes"
5 set ylabel "Time to Live"
6 plot "Vikram.txt" using 1: 5 with linespoint title "TimeDelay" lw 4
Plain Text
Tab Width: 8
Ln 1, Col 1
INS


Activities
Terminal
Aug 4 14:30
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ gnuplot vikram.plt


Activities
Document Viewer
Aug 4 14:31
1 of 1
Belesh.pdf
234.4%
Vikram Demo NS3 Lecture Graph
TimeDelay
Time to Live
nodes
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
2
4
6
8
10
12
14
16

==========

Practical 10: Program to Simulate Star Topology

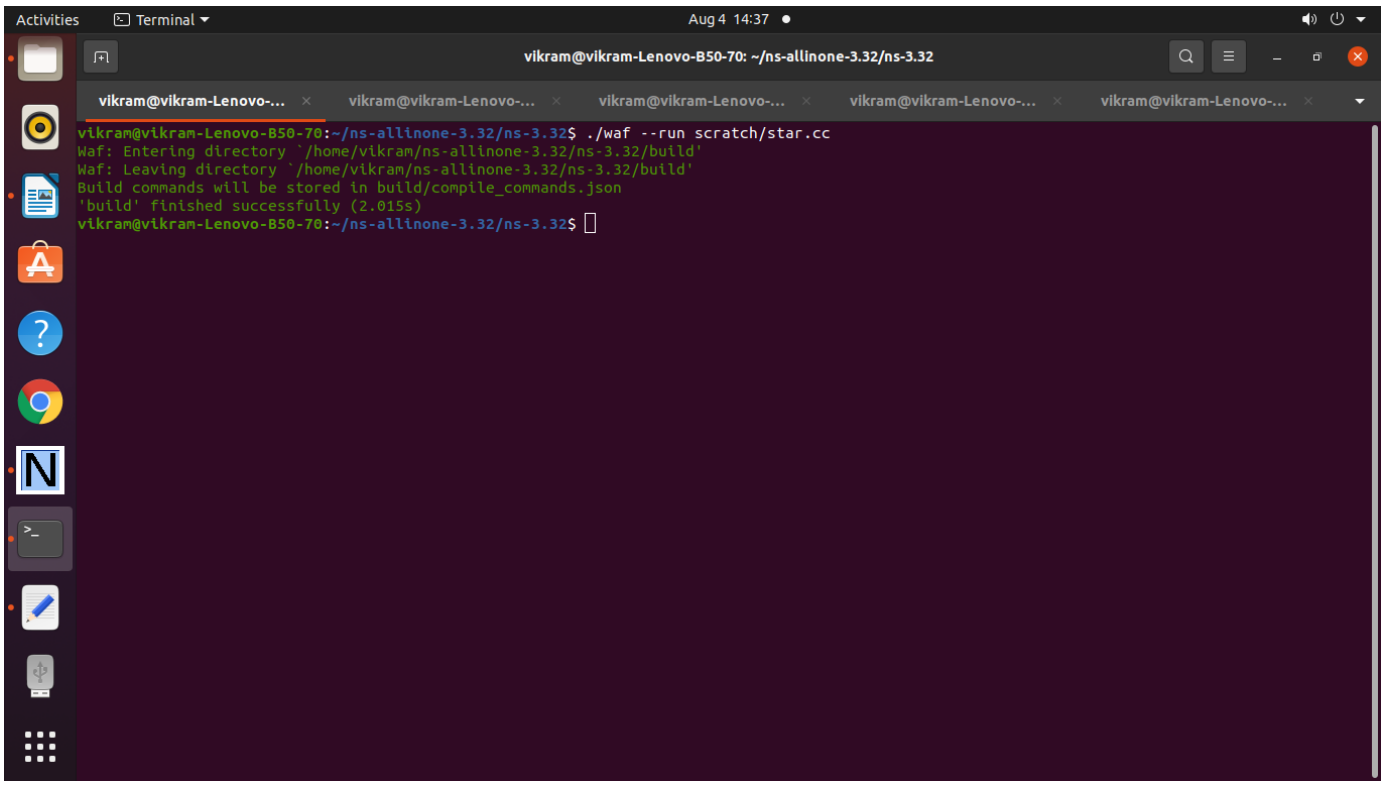
```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
   /*
   * This program is free software; you can redistribute it and/or modify
    * it under the terms of the GNU General Public License version 2 as
    * published by the Free Software Foundation;
   *
    * This program is distributed 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/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 ("StarAnimation");
```

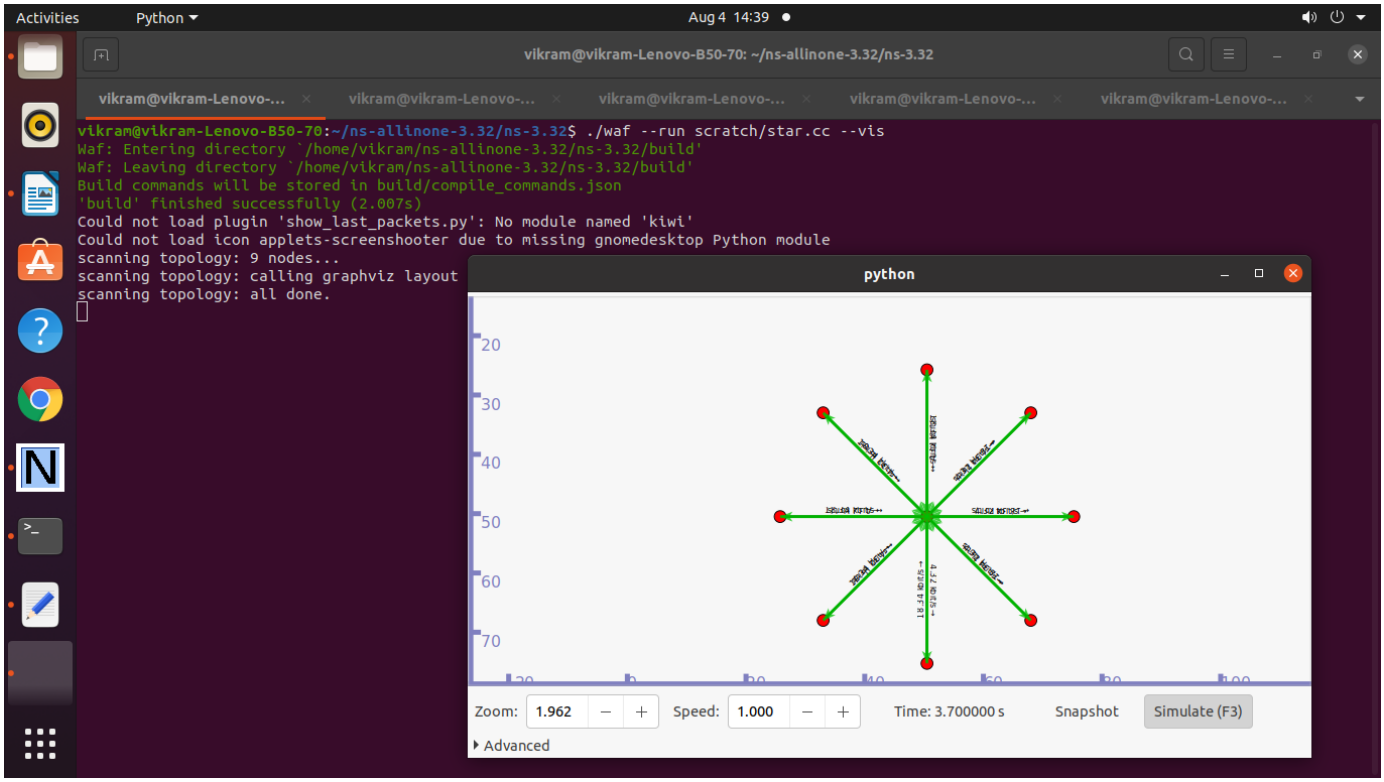
```
int
main (int argc, char *argv[])
{

 //
 // 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 line argument.
 //
uint32_t nSpokes = 8;
std::string animFile = "star-animation.xml";
 uint8_t useIpv6 = 0;
 Ipv6Address ipv6AddressBase = Ipv6Address("2001::");
Ipv6Prefix ipv6AddressPrefix = Ipv6Prefix(64);

CommandLine cmd;
 cmd.AddValue ("nSpokes", "Number of spoke nodes to place in the star",
nSpokes);
 cmd.AddValue ("animFile",  "File Name for Animation Output", animFile);
 cmd.AddValue ("useIpv6",   "use Ipv6", useIpv6);

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.");
 if (useIpv6 == 0)
  {
   star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0", "255.255.255.0"));
  }
```

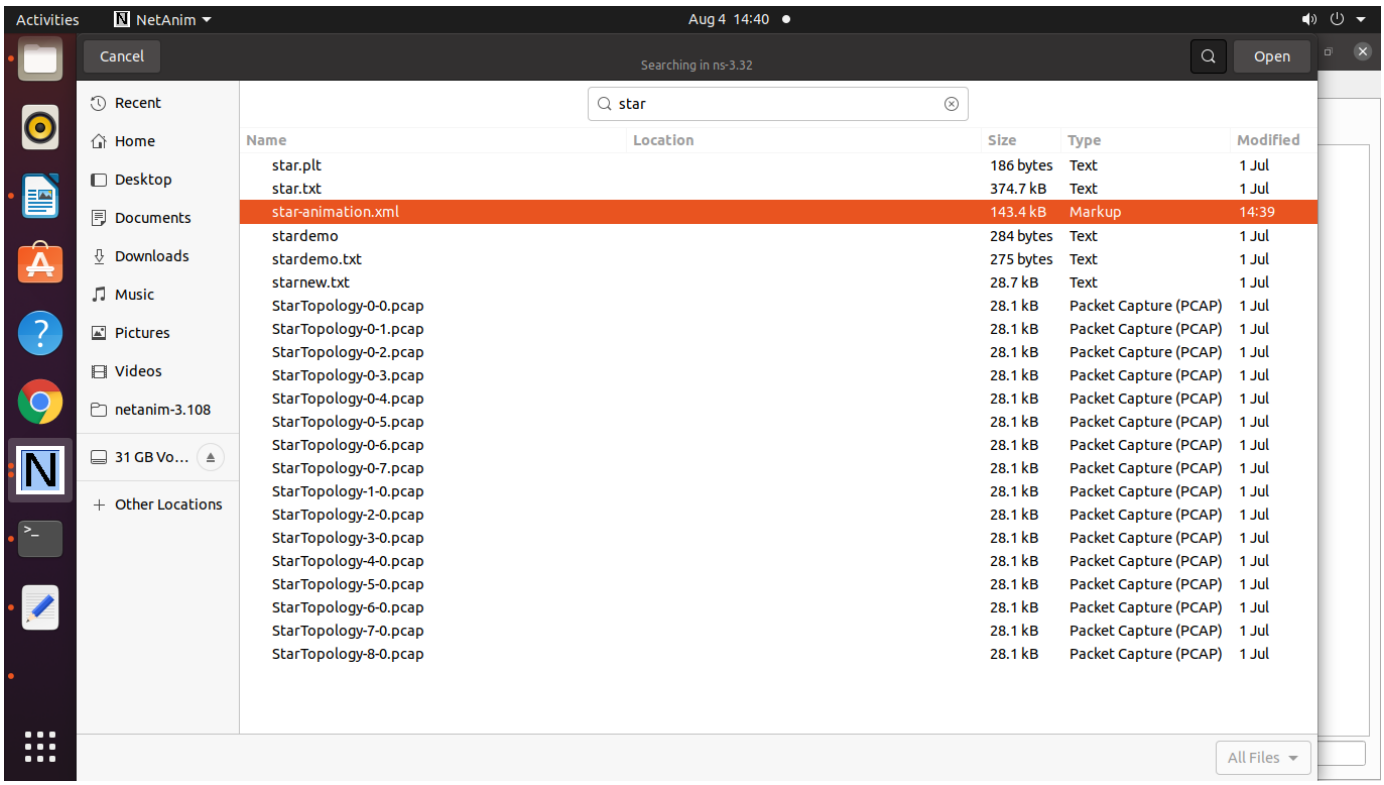
```
    else
     {
       star.AssignIpv6Addresses (ipv6AddressBase, ipv6AddressPrefix);
     }

   NS_LOG_INFO ("Create applications.");
   //
   // Create a packet sink on the star "hub" to receive packets.
    //
    uint16_t port = 50000;
    Address hubLocalAddress;
   if (useIpv6 == 0)
     {
      hubLocalAddress = InetSocketAddress (Ipv4Address::GetAny (), port);
     }
   else
     {
      hubLocalAddress = Inet6SocketAddress (Ipv6Address::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;
     if (useIpv6 == 0)
       {
         remoteAddress = AddressValue(InetSocketAddress
(star.GetHubIpv4Address (i), port));
       }
      else
```

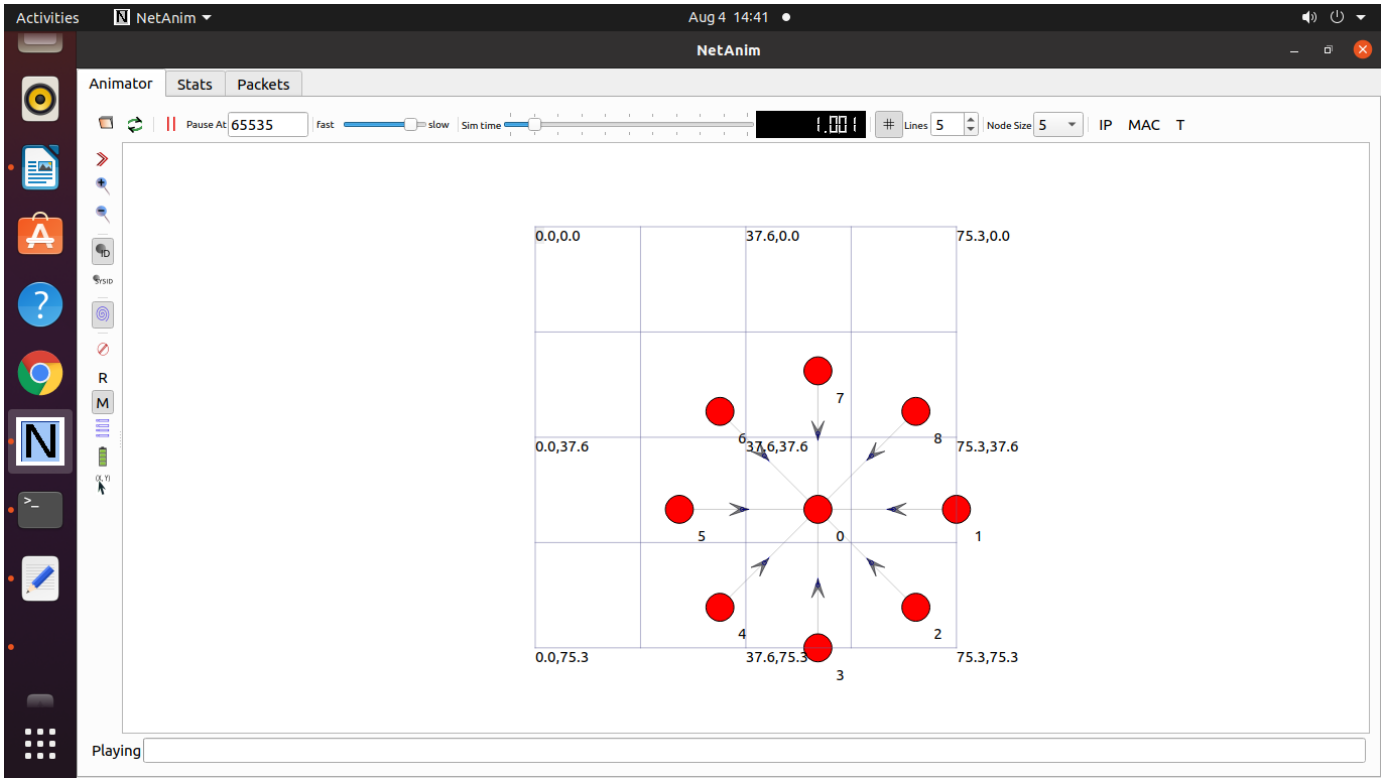
```
        {
          remoteAddress = AddressValue(Inet6SocketAddress
(star.GetHubIpv6Address (i), port));
        }
       onOffHelper.SetAttribute ("Remote", remoteAddress);
       spokeApps.Add (onOffHelper.Install (star.GetSpokeNode (i)));
     }
   spokeApps.Start (Seconds (1.0));
    spokeApps.Stop (Seconds (10.0));

    NS_LOG_INFO ("Enable static global routing.");
   //
  // Turn on global static routing so we can actually be routed across the star.
  //
  if (useIpv6 == 0)
     {
      Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
     }

    // Set the bounding box for animation
   star.BoundingBox (1, 1, 100, 100);

   // Create the animation object and configure for specified output
   AnimationInterface anim (animFile);

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

    return 0;
 }
```

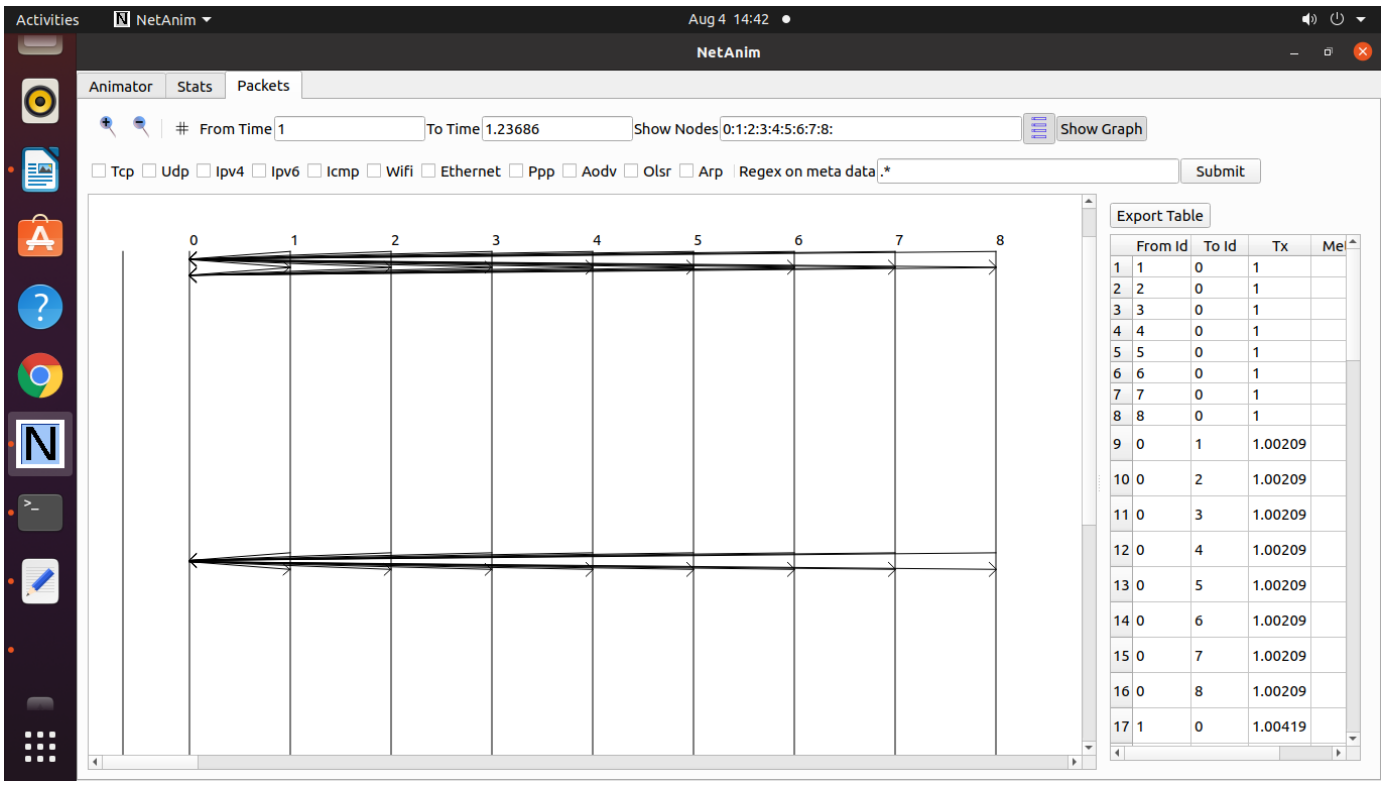
Output :


Activities
Terminal
Aug 4 14:37
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/star.cc
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.015s)
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$


Activities
Python
Aug 4 14:39
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-...
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/star.cc --vis
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.007s)
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.
python
20
30
40
50
60
70
Zoom: 1.962
Speed: 1.000
Time: 3.700000 s
Snapshot
Simulate (F3)
Advanced

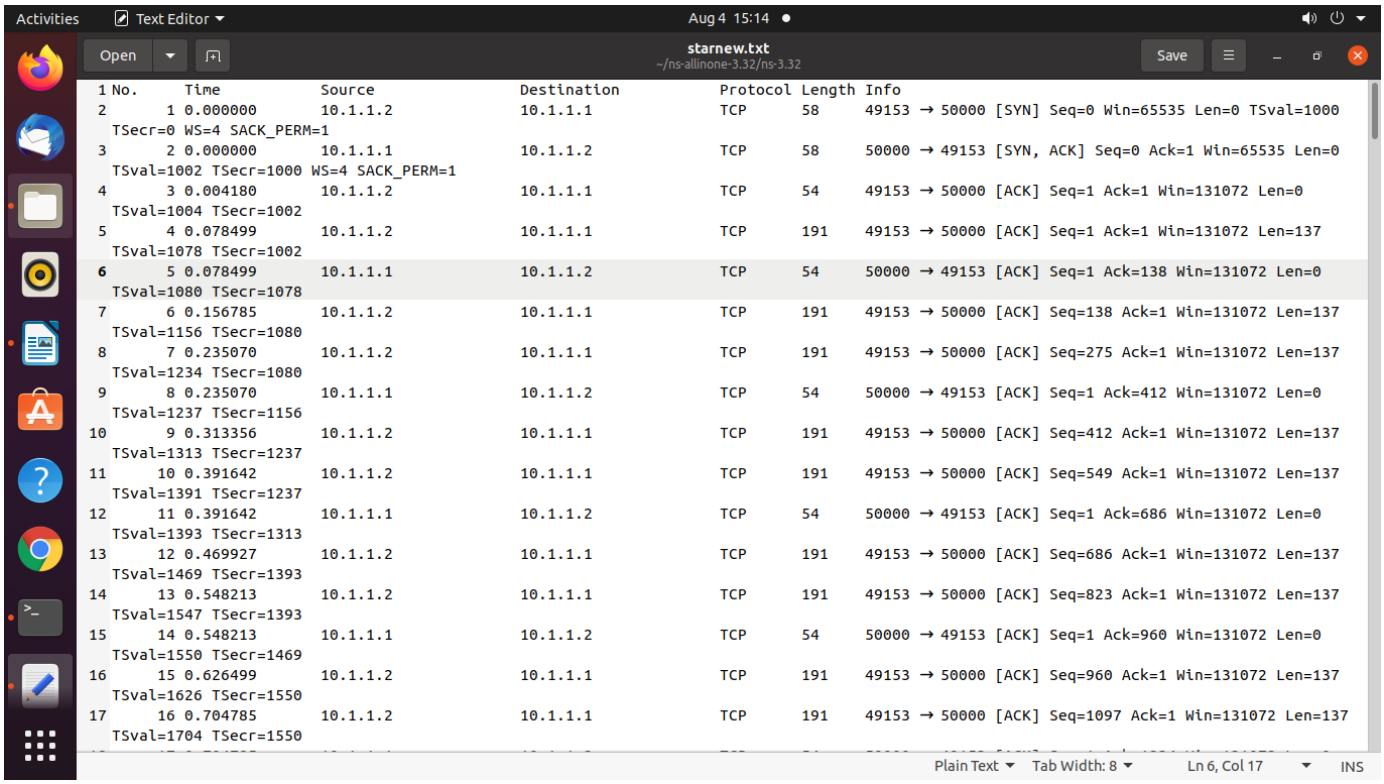

Activities
NetAnim
Aug 4 14:40
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
31 GB Vo...
Other Locations
star
Name
Location
Size
Type
Modified
star.plt 186 bytes Text 1 Jul
star.txt 374.7 kB Text 1 Jul
star-animation.xml 143.4 kB Markup 14:39
stardemo 284 bytes Text 1 Jul
stardemo.txt 275 bytes Text 1 Jul
starnew.txt 28.7 kB Text 1 Jul
StarTopology-0-0.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-0-1.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-0-2.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-0-3.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-0-4.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-0-5.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-0-6.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-0-7.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-1-0.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-2-0.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-3-0.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-4-0.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-5-0.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-6-0.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-7-0.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
StarTopology-8-0.pcap 28.1 kB Packet Capture (PCAP) 1 Jul
All Files


Activities
NetAnim
Aug 4 14:41
NetAnim
Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
Lines 5
Node Size 5
IP
MAC
T
0.0,0.0
37.6,0.0
75.3,0.0
0.0,37.6
37.6,37.6
75.3,37.6
0.0,75.3
37.6,75.3
75.3,75.3
0
1
2
3
4
5
6
7
8
Playing



To Plot the graph consider the following files

starnew.txt



Activities Text Editor Aug 4 15:15

Open star.plt ~/ns-allinone-3.32/ns-3.32 Save

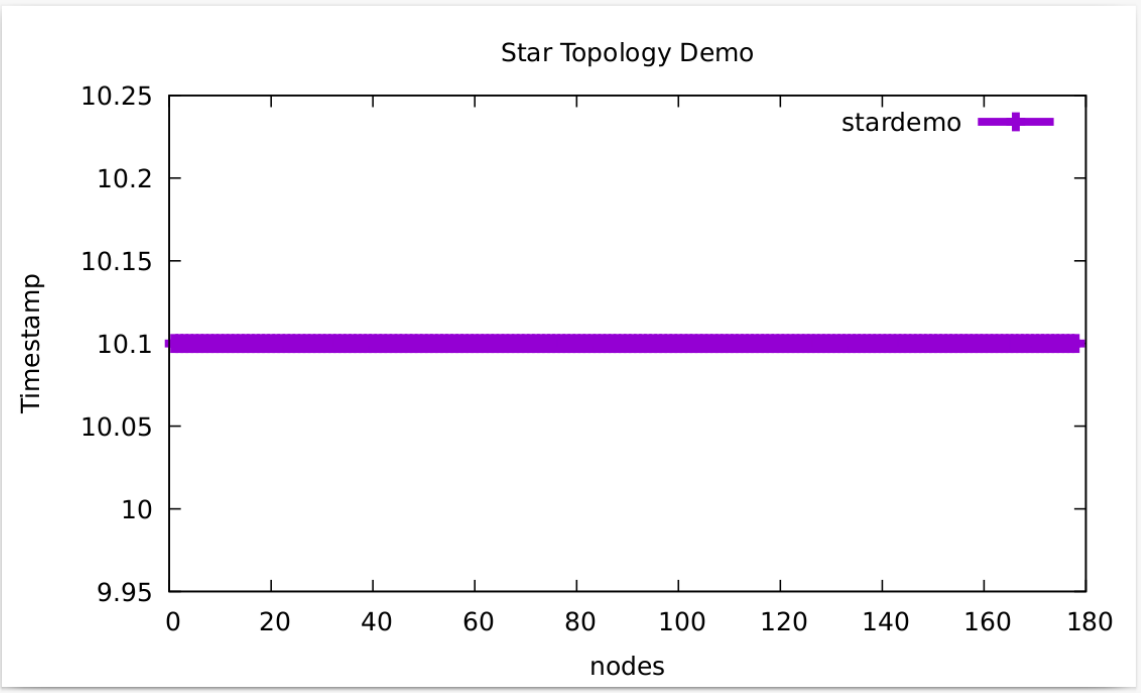
starnew.txt star.plt

```
1 set terminal pdf
2 set output "Komal.pdf"
3 set title   "Star Topology Demo"
4 set xlabel  "nodes"
5 set ylabel  "Timestamp"
6 plot "starnew.txt" using 1: 4 with linespoint title "stardemo" lw 5
7
```

Plain Text Tab Width: 8 Ln 6, Col 14 INS

Activities Document Viewer Aug 4 15:15

1 of 1 Komal.pdf 234.4%



Practical 11 : Program to simulate UDP Server – Client communication.

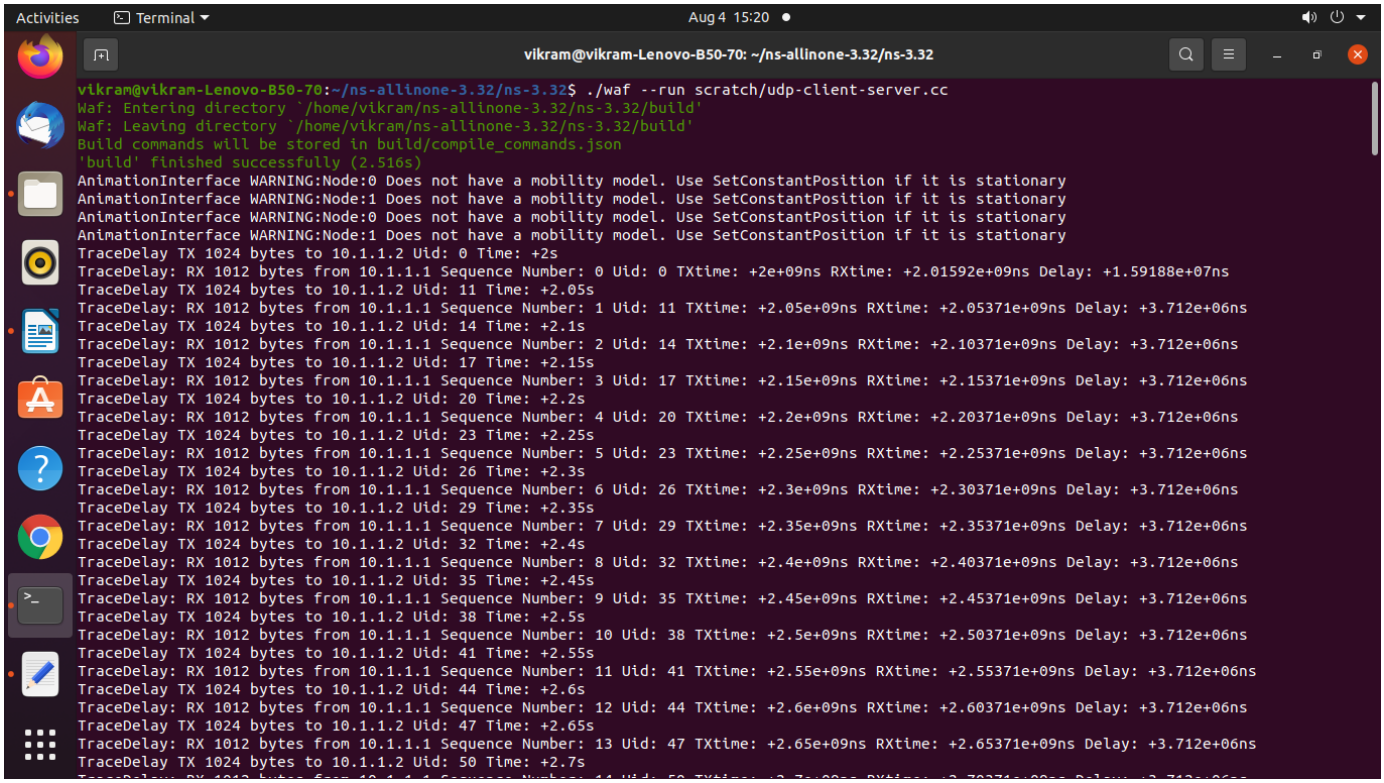
```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the 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
//       |     |
//       =======
//         LAN
//
// - UDP flows from n0 to n1

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

//netAnimation

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

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("UdpClientServerExample");

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

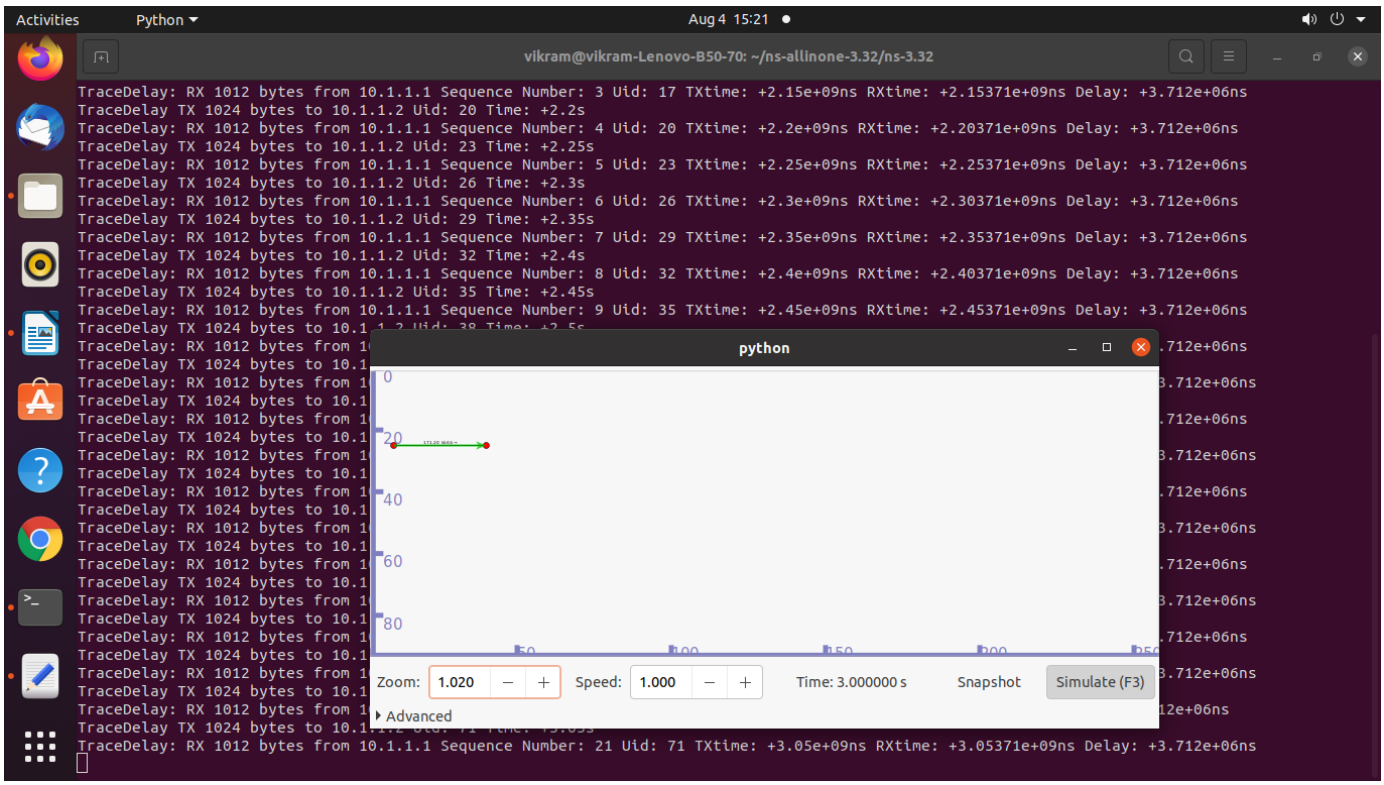
```
{
//
// Enable logging for UdpClient and
//
  LogComponentEnable ("UdpClient", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);

  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 (2);

  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);

//
// 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));
    }
```

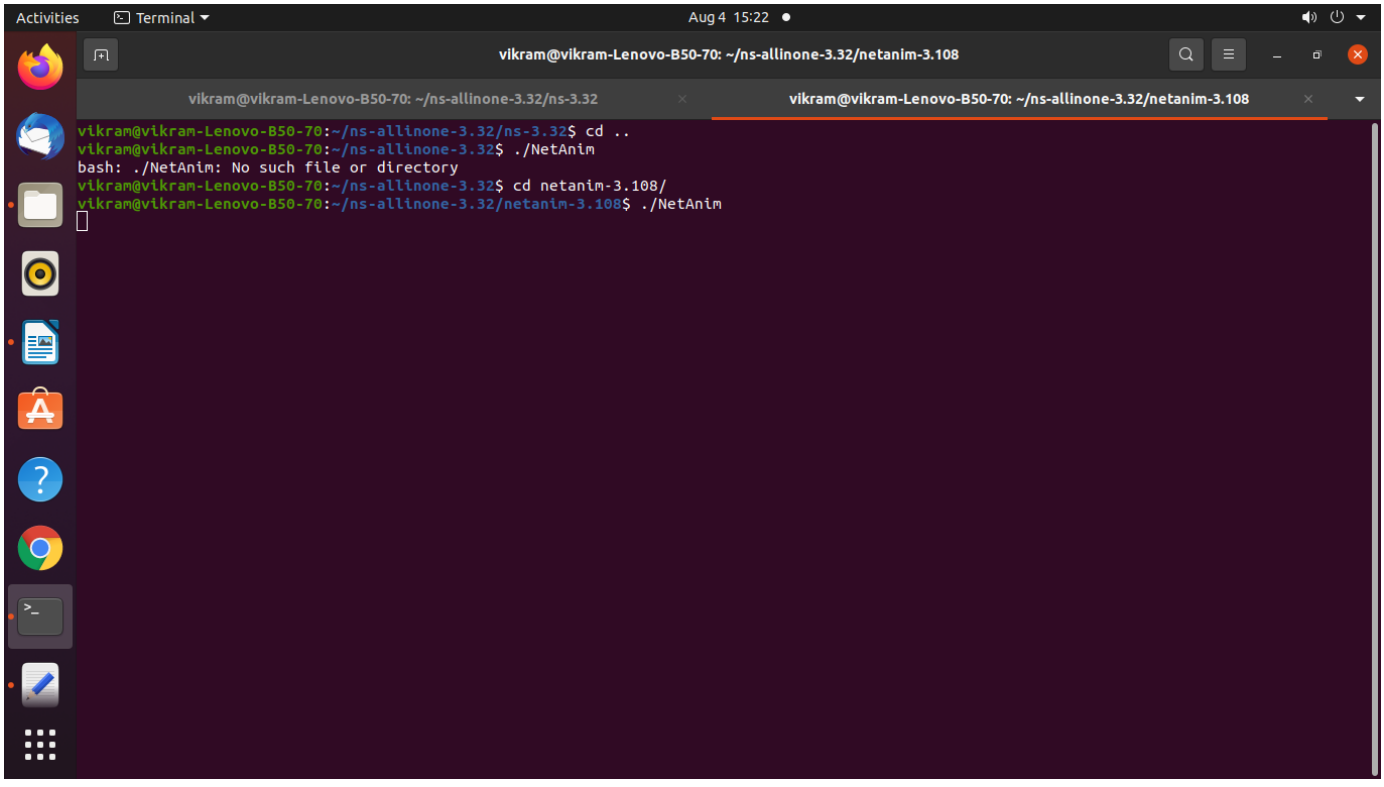
```
  else
    {
      Ipv6AddressHelper ipv6;
      ipv6.SetBase ("2001:0000:f00d:cafe::", Ipv6Prefix (64));
      Ipv6InterfaceContainer i6 = ipv6.Assign (d);
      serverAddress = Address(i6.GetAddress (1,1));
    }

  NS_LOG_INFO ("Create Applications.");
//
// Create one udpServer applications on node one.
//
  uint16_t port = 4000;
  UdpServerHelper server (port);
  ApplicationContainer apps = server.Install (n.Get (1));
  apps.Start (Seconds (1.0));
  apps.Stop (Seconds (10.0));

//
// Create one UdpClient application to send UDP datagrams from node zero to
// node one.
//
  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));

  AnimationInterface anim("udp-cs.xml");
AnimationInterface::SetConstantPosition (n.Get(0), 10, 25);
AnimationInterface ::SetConstantPosition(n.Get(1), 40,25);
anim.EnablePacketMetadata(true);
csma.EnablePcapAll("udp-cs");



//
// Now, do the actual simulation.
//
  NS_LOG_INFO ("Run Simulation.");
```

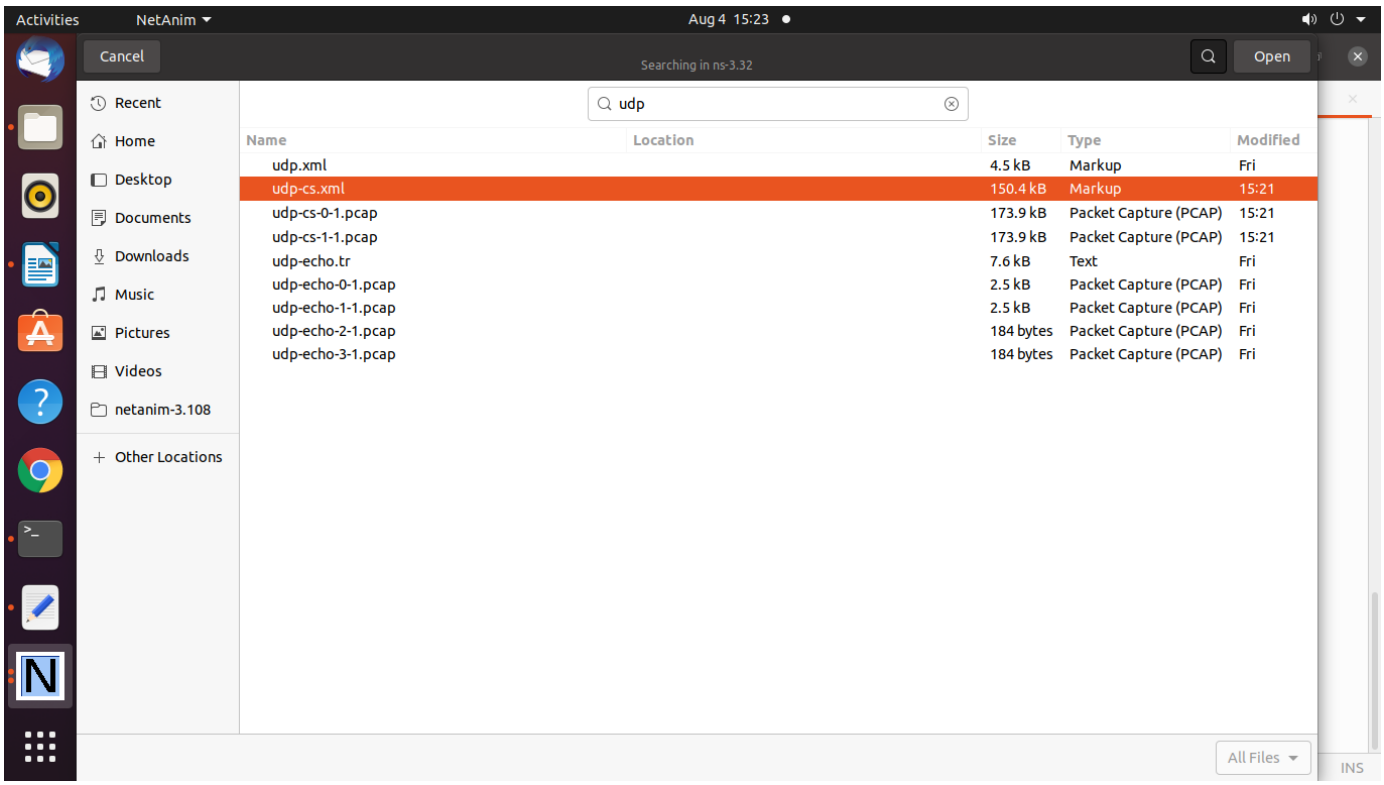
```
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");
}
```

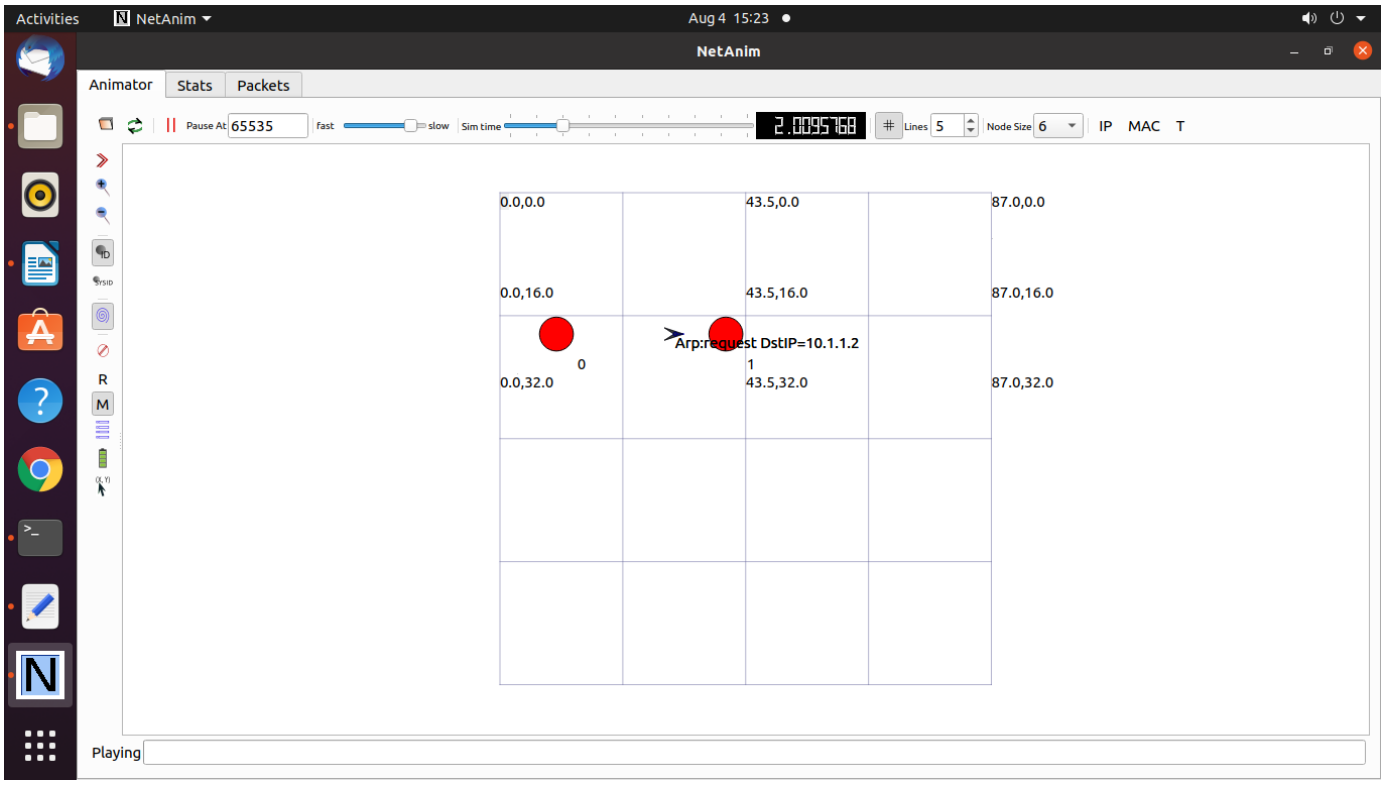
Output :

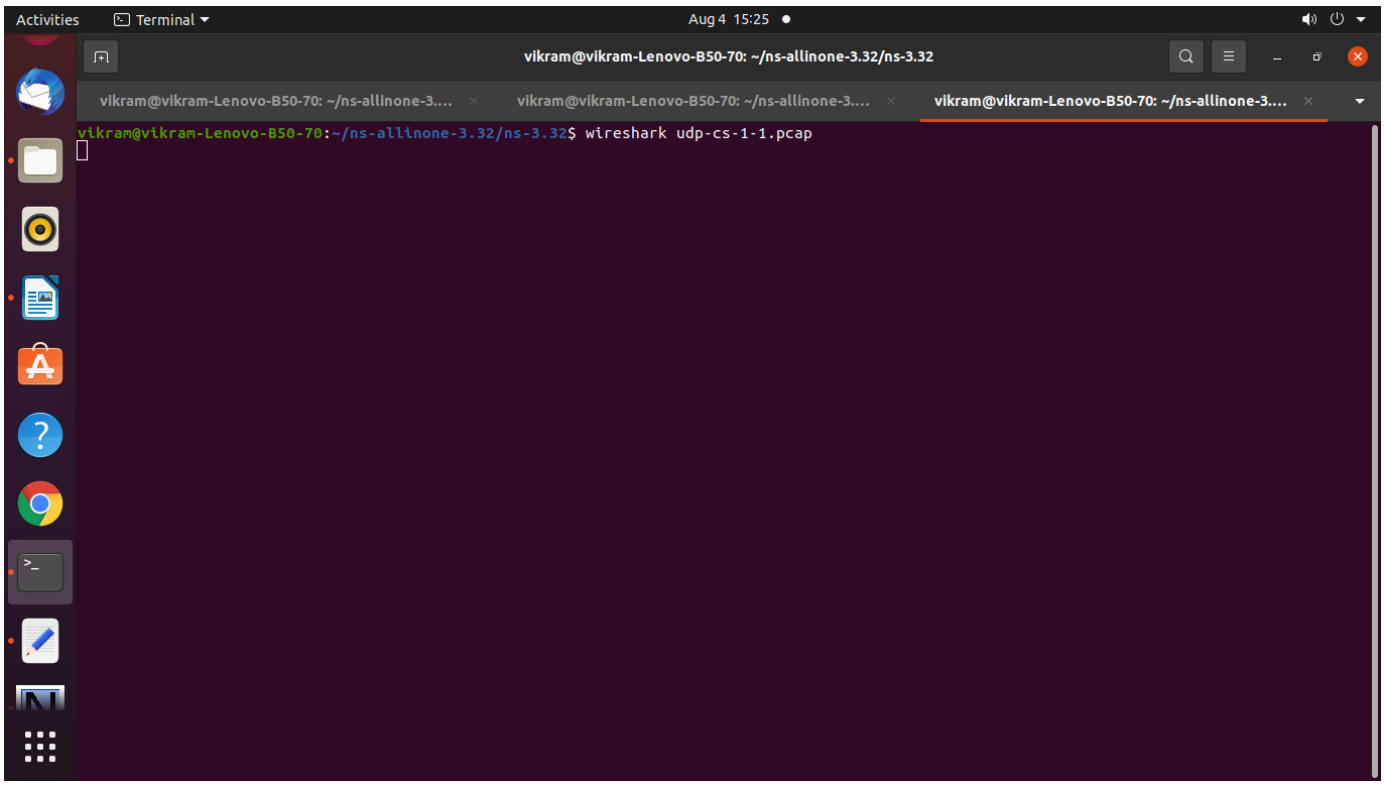


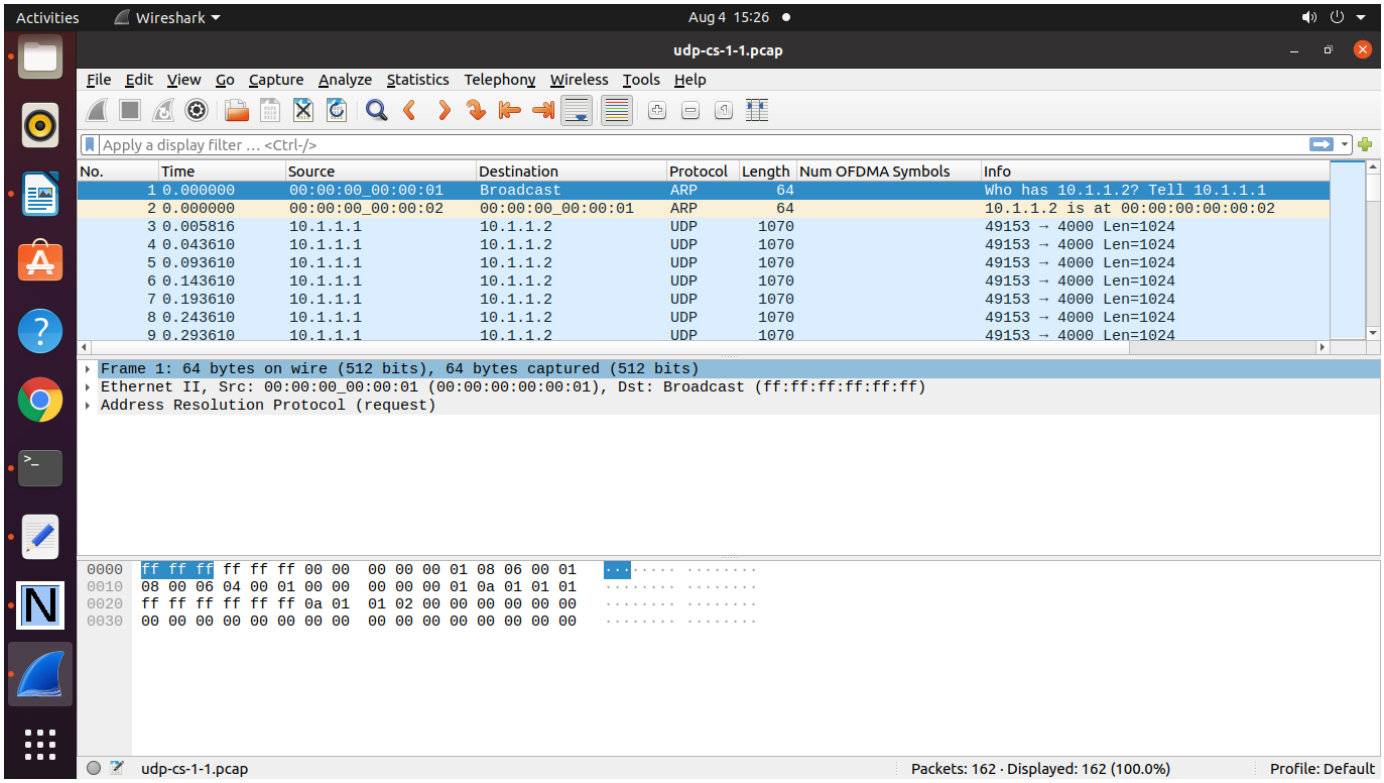

Activities
Python
Aug 4 15:21
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 3 Uid: 17 TXtime: +2.15e+09ns RXtime: +2.15371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 20 Time: +2.2s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 4 Uid: 20 TXtime: +2.2e+09ns RXtime: +2.20371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 23 Time: +2.25s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 5 Uid: 23 TXtime: +2.25e+09ns RXtime: +2.25371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 26 Time: +2.3s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 6 Uid: 26 TXtime: +2.3e+09ns RXtime: +2.30371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 29 Time: +2.35s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 7 Uid: 29 TXtime: +2.35e+09ns RXtime: +2.35371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 32 Time: +2.4s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 8 Uid: 32 TXtime: +2.4e+09ns RXtime: +2.40371e+09ns Delay: +3.712e+06ns
TraceDelay TX 1024 bytes to 10.1.1.2 Uid: 35 Time: +2.45s
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 9 Uid: 35 TXtime: +2.45e+09ns RXtime: +2.45371e+09ns Delay: +3.712e+06ns
python
0
20
40
60
80
Zoom: 1.020
Speed: 1.000
Time: 3.000000 s
Snapshot
Simulate (F3)
Advanced
TraceDelay: RX 1012 bytes from 10.1.1.1 Sequence Number: 21 Uid: 71 TXtime: +3.05e+09ns RXtime: +3.05371e+09ns Delay: +3.712e+06ns

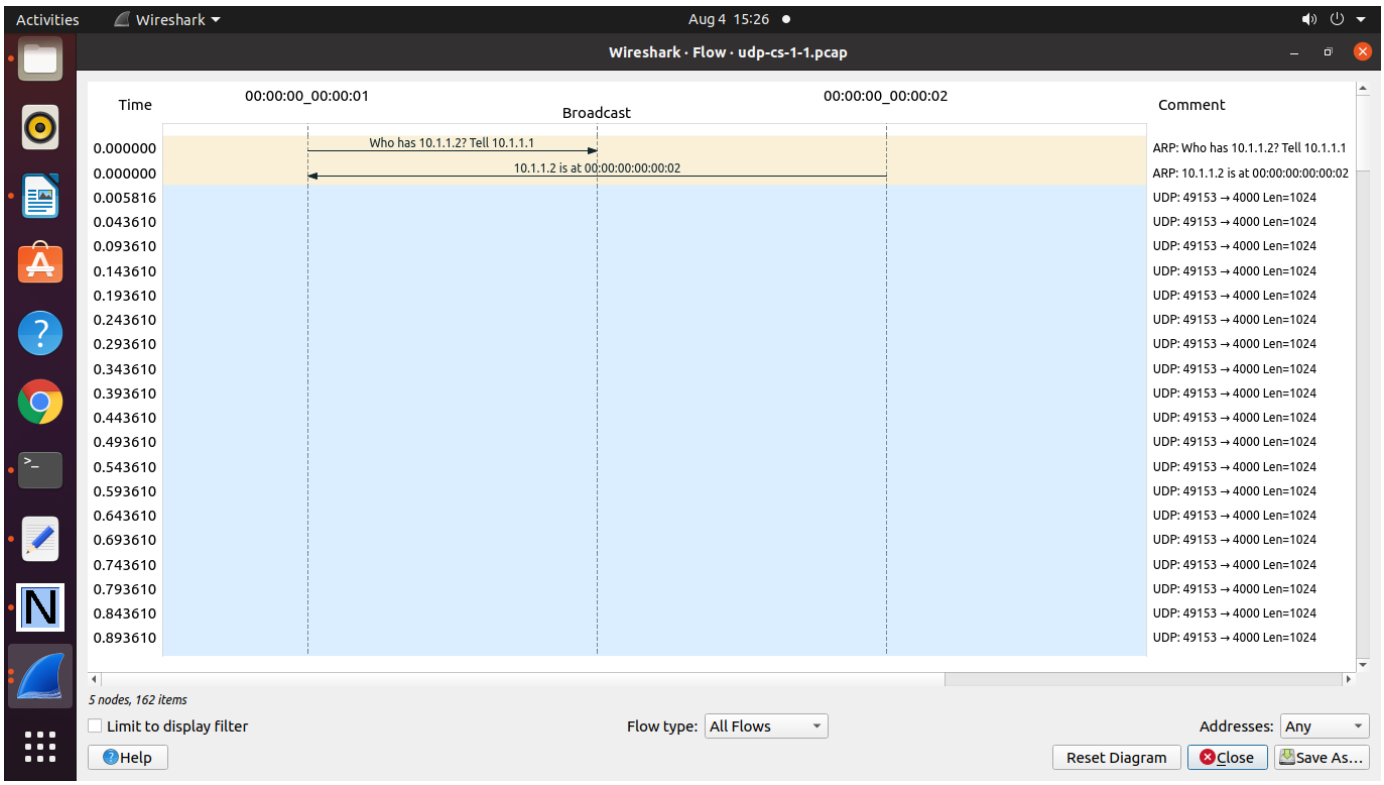

Activities
Terminal
Aug 4 15:22
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/netanim-3.108
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/netanim-3.108
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ cd ..
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32$ ./NetAnim
bash: ./NetAnim: No such file or directory
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32$ cd netanim-3.108/
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/netanim-3.108$ ./NetAnim

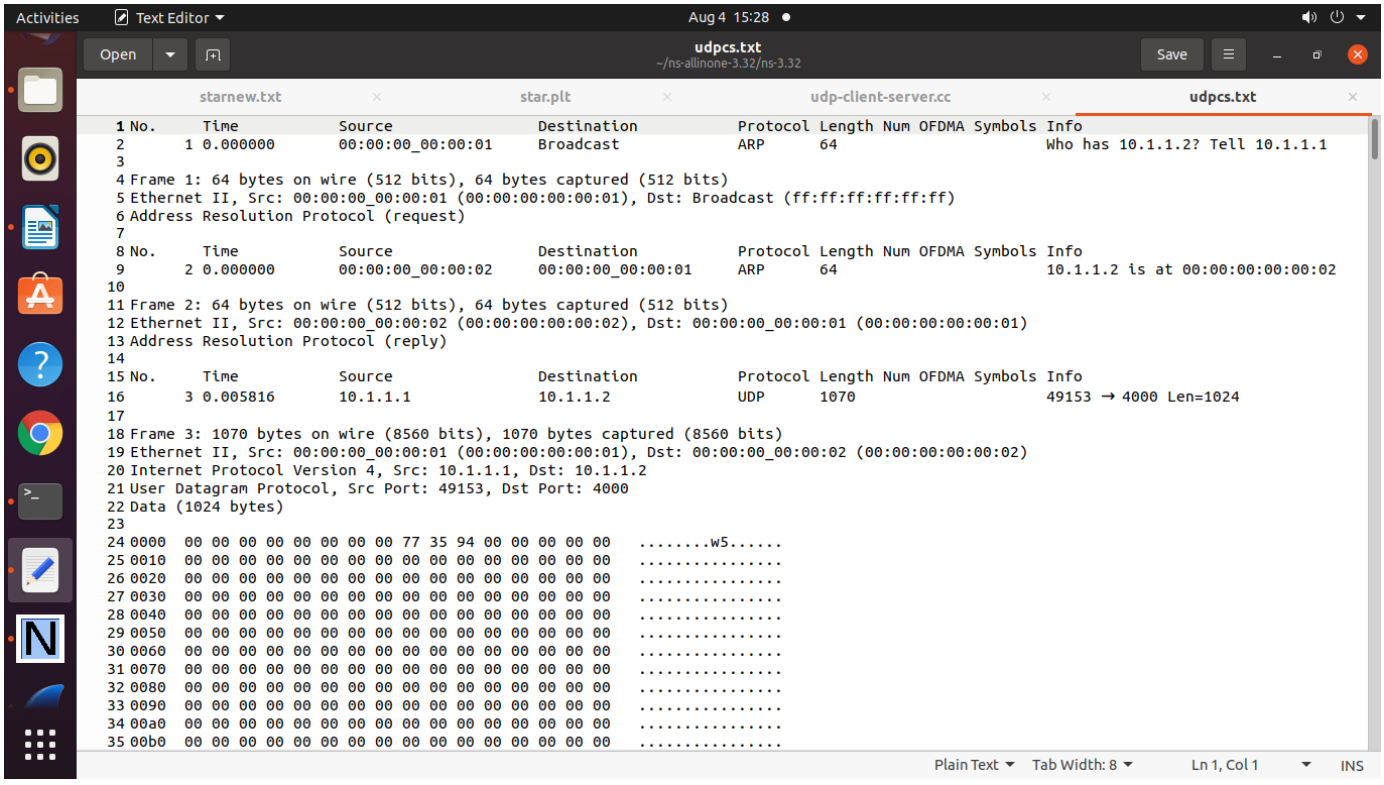

Activities
NetAnim
Aug 4 15:23
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
udp
Name
Location
Size
Type
Modified
udp.xml 4.5 kB Markup Fri
udp-cs.xml 150.4 kB Markup 15:21
udp-cs-0-1.pcap 173.9 kB Packet Capture (PCAP) 15:21
udp-cs-1-1.pcap 173.9 kB Packet Capture (PCAP) 15:21
udp-echo.tr 7.6 kB Text Fri
udp-echo-0-1.pcap 2.5 kB Packet Capture (PCAP) Fri
udp-echo-1-1.pcap 2.5 kB Packet Capture (PCAP) Fri
udp-echo-2-1.pcap 184 bytes Packet Capture (PCAP) Fri
udp-echo-3-1.pcap 184 bytes Packet Capture (PCAP) Fri
All Files
INS


Activities
NetAnim
Aug 4 15:23
NetAnim
Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
2.0095768
Lines 5
Node Size 6
IP
MAC
T
0.0,0.0
43.5,0.0
87.0,0.0
0.0,16.0
43.5,16.0
87.0,16.0
Arp:request DstIP=10.1.1.2
0
1
0.0,32.0
43.5,32.0
87.0,32.0
Playing


Activities
Terminal
Aug 4 15:25
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ wireshark udp-cs-1-1.pcap


Activities
Wireshark
Aug 4 15:26
udp-cs-1-1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Num OFDMA Symbols Info
1 0.000000 00:00:00_00:00:01 Broadcast ARP 64 Who has 10.1.1.2? Tell 10.1.1.1
2 0.000000 00:00:00_00:00:02 00:00:00_00:00:01 ARP 64 10.1.1.2 is at 00:00:00:00:00:02
3 0.005816 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=1024
4 0.043610 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=1024
5 0.093610 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=1024
6 0.143610 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=1024
7 0.193610 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=1024
8 0.243610 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=1024
9 0.293610 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=1024
Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
Ethernet II, Src: 00:00:00_00:00:01 (00:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)
0000 ff ff ff ff ff ff 00 00 00 00 00 01 08 06 00 01
0010 08 00 06 04 00 01 00 00 00 00 00 01 0a 01 01 01
0020 ff ff ff ff ff ff 0a 01 01 02 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
udp-cs-1-1.pcap
Packets: 162 · Displayed: 162 (100.0%)
Profile: Default


Activities
Wireshark
Aug 4 15:26
Wireshark · Flow · udp-cs-1-1.pcap
Time
00:00:00_00:00:01
Broadcast
00:00:00_00:00:02
Comment
Who has 10.1.1.2? Tell 10.1.1.1
10.1.1.2 is at 00:00:00:00:00:02
0.000000 ARP: Who has 10.1.1.2? Tell 10.1.1.1
0.000000 ARP: 10.1.1.2 is at 00:00:00:00:00:02
0.005816 UDP: 49153 → 4000 Len=1024
0.043610 UDP: 49153 → 4000 Len=1024
0.093610 UDP: 49153 → 4000 Len=1024
0.143610 UDP: 49153 → 4000 Len=1024
0.193610 UDP: 49153 → 4000 Len=1024
0.243610 UDP: 49153 → 4000 Len=1024
0.293610 UDP: 49153 → 4000 Len=1024
0.343610 UDP: 49153 → 4000 Len=1024
0.393610 UDP: 49153 → 4000 Len=1024
0.443610 UDP: 49153 → 4000 Len=1024
0.493610 UDP: 49153 → 4000 Len=1024
0.543610 UDP: 49153 → 4000 Len=1024
0.593610 UDP: 49153 → 4000 Len=1024
0.643610 UDP: 49153 → 4000 Len=1024
0.693610 UDP: 49153 → 4000 Len=1024
0.743610 UDP: 49153 → 4000 Len=1024
0.793610 UDP: 49153 → 4000 Len=1024
0.843610 UDP: 49153 → 4000 Len=1024
0.893610 UDP: 49153 → 4000 Len=1024
5 nodes, 162 items
Limit to display filter
Flow type: All Flows
Addresses: Any
Help
Reset Diagram
Close
Save As...


Activities
Text Editor
Aug 4 15:28
Open
udpcs.txt
~/ns-allinone-3.32/ns-3.32
Save
starnew.txt
star.plt
udp-client-server.cc
udpcs.txt
1 No. Time Source Destination Protocol Length Num OFDMA Symbols Info
2 1 0.000000 00:00:00_00:00:01 Broadcast ARP 64 Who has 10.1.1.2? Tell 10.1.1.1
3
4 Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
5 Ethernet II, Src: 00:00:00_00:00:01 (00:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
6 Address Resolution Protocol (request)
7
8 No. Time Source Destination Protocol Length Num OFDMA Symbols Info
9 2 0.000000 00:00:00_00:00:02 00:00:00_00:00:01 ARP 64 10.1.1.2 is at 00:00:00:00:00:02
10
11 Frame 2: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
12 Ethernet II, Src: 00:00:00_00:00:02 (00:00:00:00:00:02), Dst: 00:00:00_00:00:01 (00:00:00:00:00:01)
13 Address Resolution Protocol (reply)
14
15 No. Time Source Destination Protocol Length Num OFDMA Symbols Info
16 3 0.005816 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=1024
17
18 Frame 3: 1070 bytes on wire (8560 bits), 1070 bytes captured (8560 bits)
19 Ethernet II, Src: 00:00:00_00:00:01 (00:00:00:00:00:01), Dst: 00:00:00_00:00:02 (00:00:00:00:00:02)
20 Internet Protocol Version 4, Src: 10.1.1.1, Dst: 10.1.1.2
21 User Datagram Protocol, Src Port: 49153, Dst Port: 4000
22 Data (1024 bytes)
23
24 0000 00 00 00 00 00 00 00 00 77 35 94 00 00 00 00 00 ....w5......
25 0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
26 0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
27 0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
28 0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
29 0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
30 0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
31 0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
32 0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
33 0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
34 00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
35 00b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

==========

Practical 12 : Program to simulatr Bus Toplogy using UDP Protocol

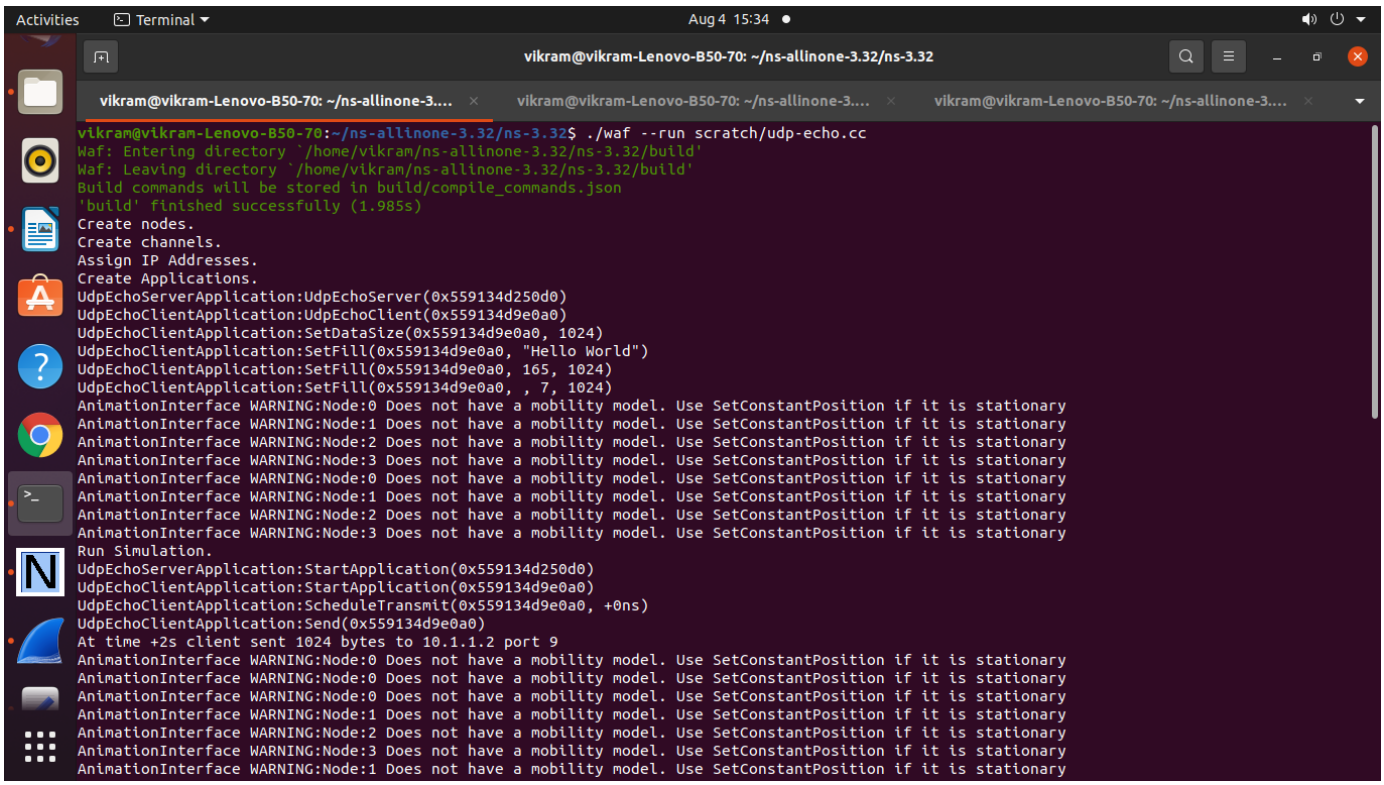
```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the 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
//     |    |    |    |
//     =================
```

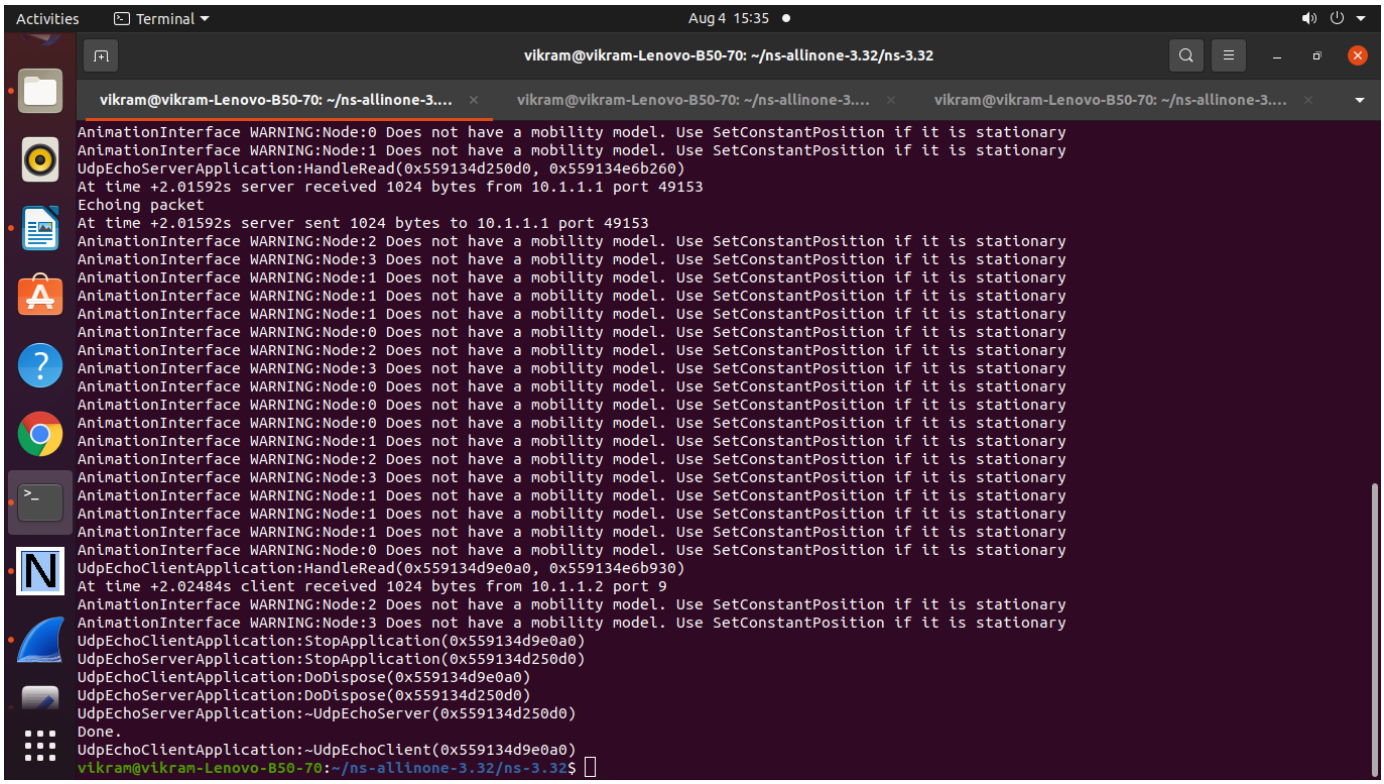
```
//           LAN
//
// - UDP flows from n0 to n1 and back
// - DropTail queues
// - Tracing of queues and packet receptions to file "udp-echo.tr"

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

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("UdpEchoExample");

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 1
  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;
  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;
```

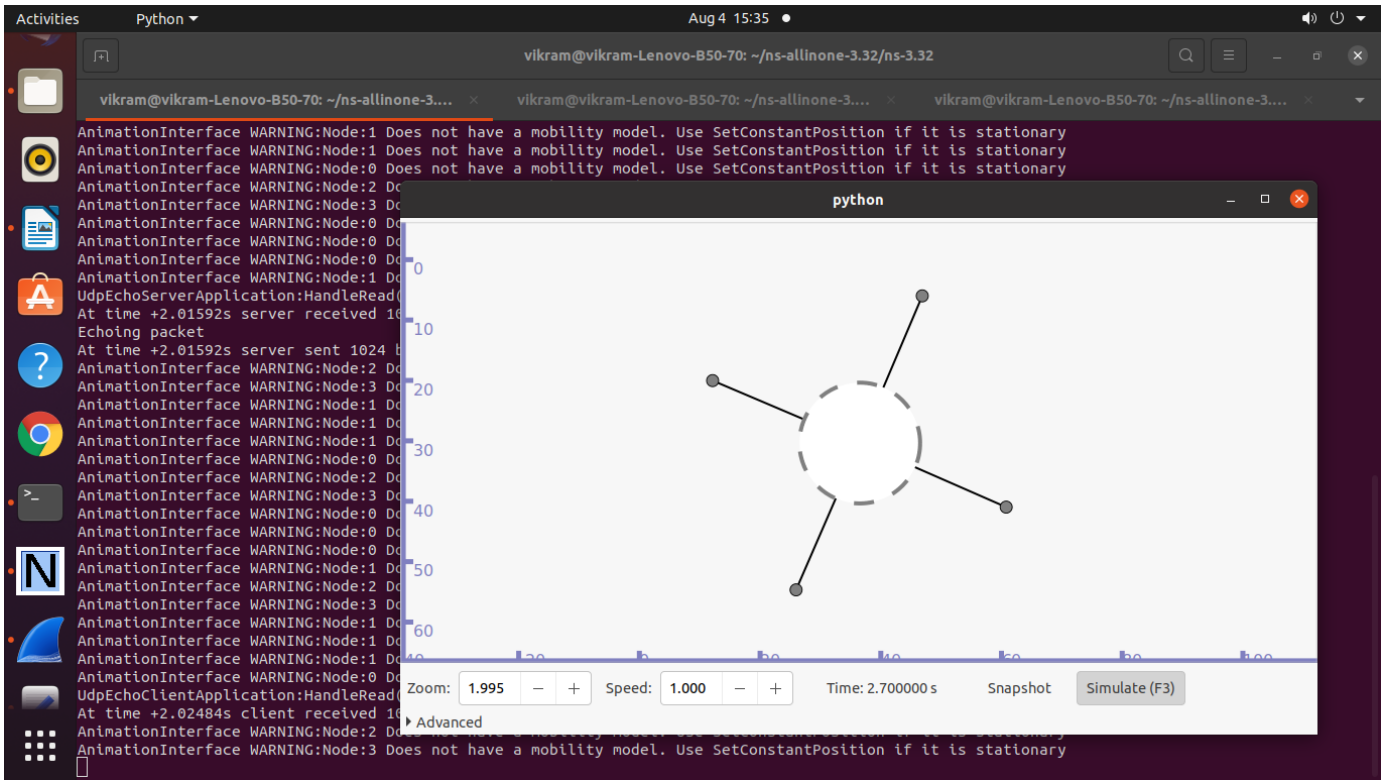
```
  n.Create (4);

  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);

//
// 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));

//
```

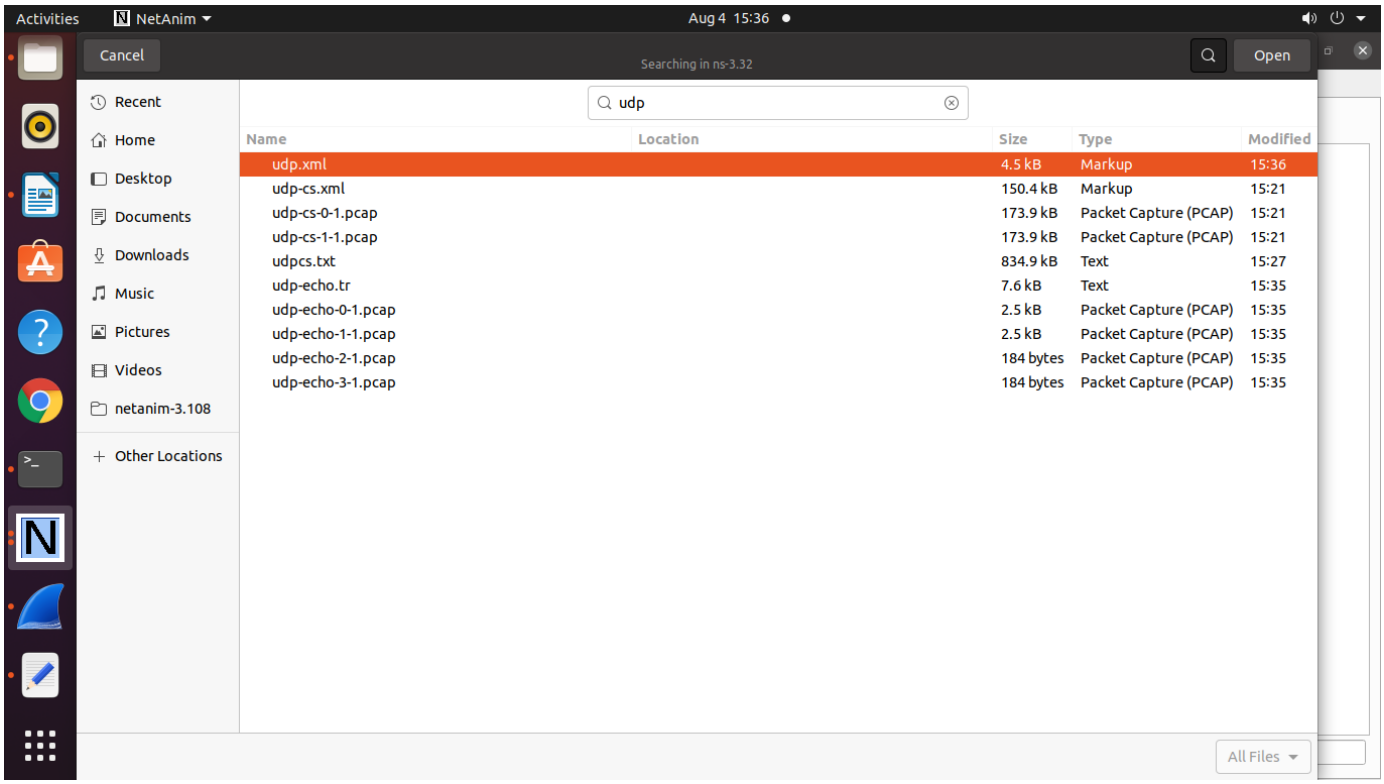
```
// Create a UdpEchoClient application to send UDP datagrams from node zero to
// node one.
//
  uint32_t packetSize = 1024;
  uint32_t maxPacketCount = 1;
  Time interPacketInterval = Seconds (1.);
  UdpEchoClientHelper client (serverAddress, port);
  client.SetAttribute ("MaxPackets", UintegerValue (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 1
//
// 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
   AnimationInterface anim("udp.xml");
  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.");
}
```

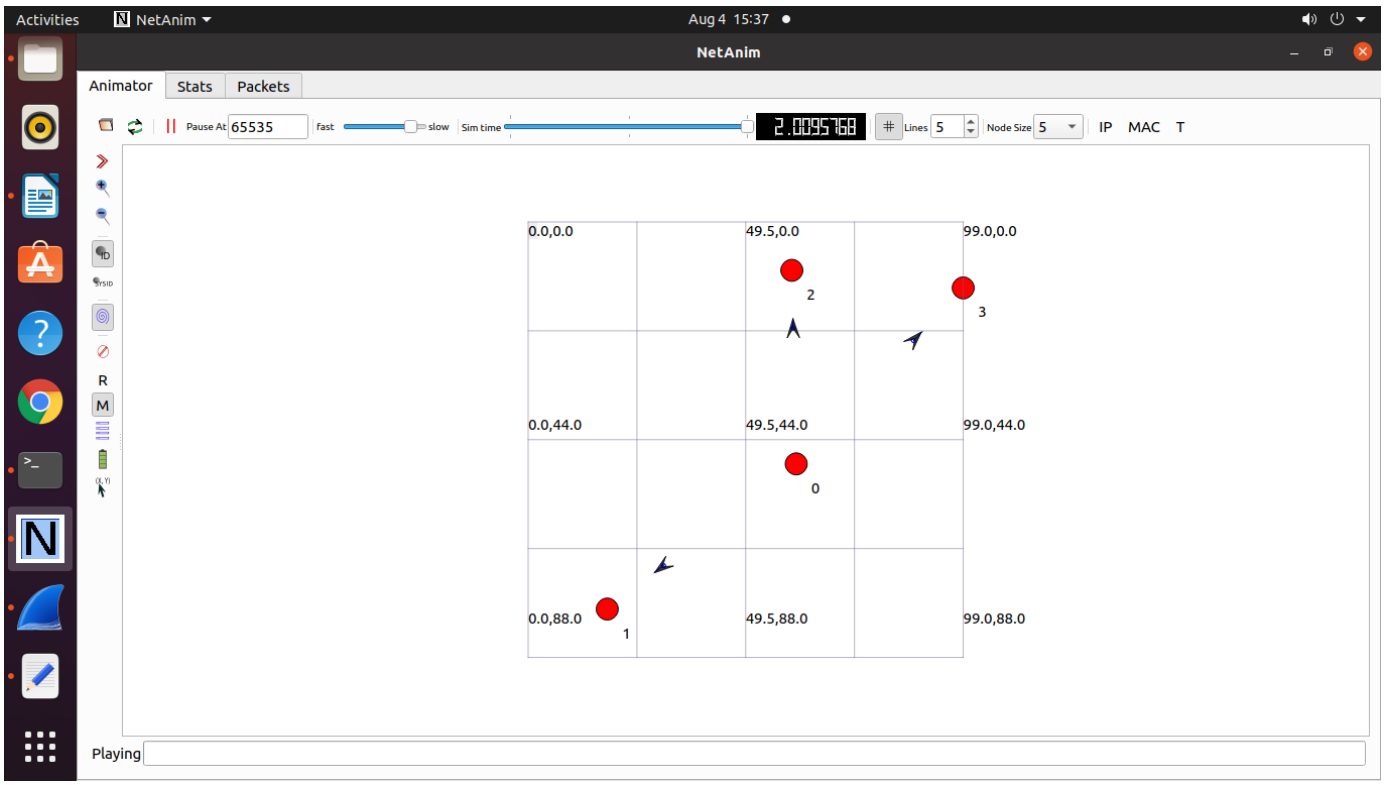
Output :

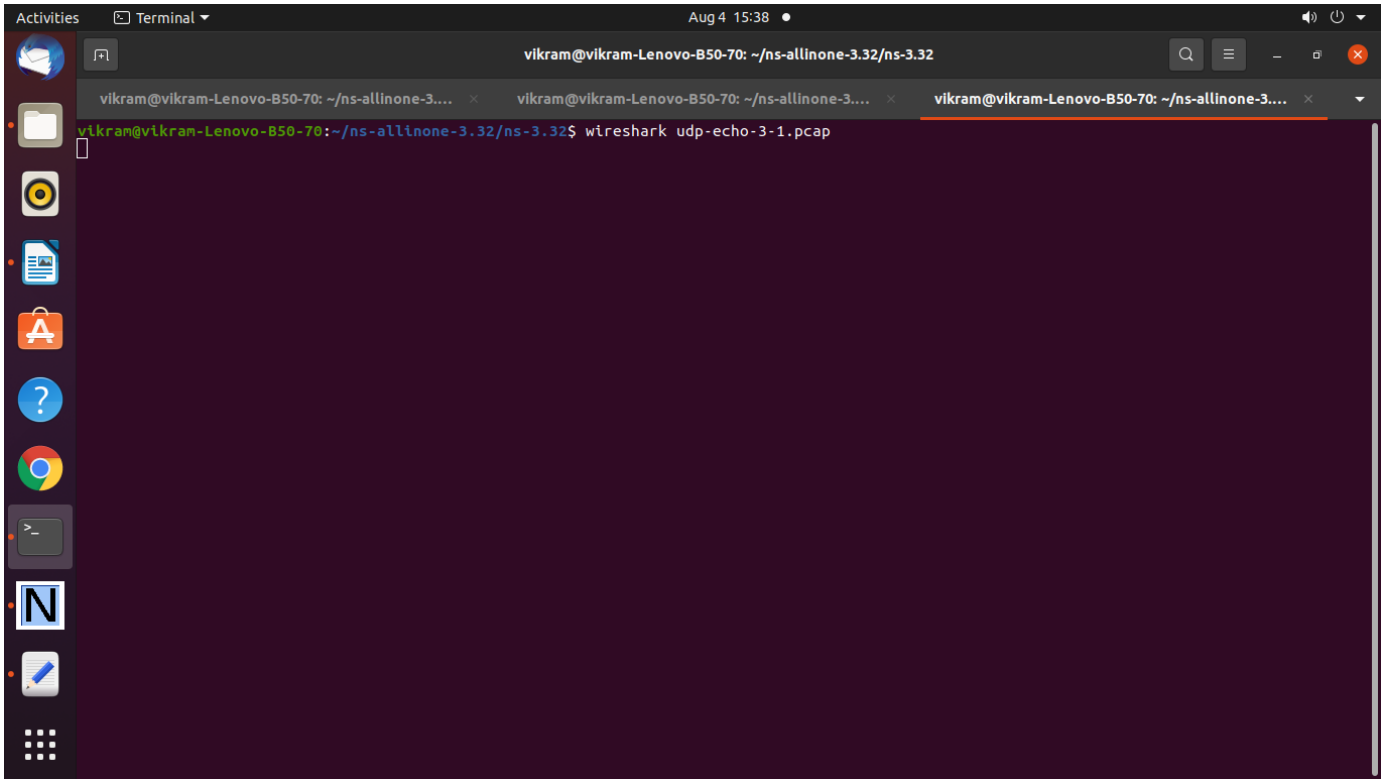

Activities
Terminal
Aug 4 15:34
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/udp-echo.cc
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.985s)
Create nodes.
Create channels.
Assign IP Addresses.
Create Applications.
UdpEchoServerApplication:UdpEchoServer(0x559134d250d0)
UdpEchoClientApplication:UdpEchoClient(0x559134d9e0a0)
UdpEchoClientApplication:SetDataSize(0x559134d9e0a0, 1024)
UdpEchoClientApplication:SetFill(0x559134d9e0a0, "Hello World")
UdpEchoClientApplication:SetFill(0x559134d9e0a0, 165, 1024)
UdpEchoClientApplication:SetFill(0x559134d9e0a0, , 7, 1024)
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
Run Simulation.
UdpEchoServerApplication:StartApplication(0x559134d250d0)
UdpEchoClientApplication:StartApplication(0x559134d9e0a0)
UdpEchoClientApplication:ScheduleTransmit(0x559134d9e0a0, +0ns)
UdpEchoClientApplication:Send(0x559134d9e0a0)
At time +2s client sent 1024 bytes to 10.1.1.2 port 9
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary

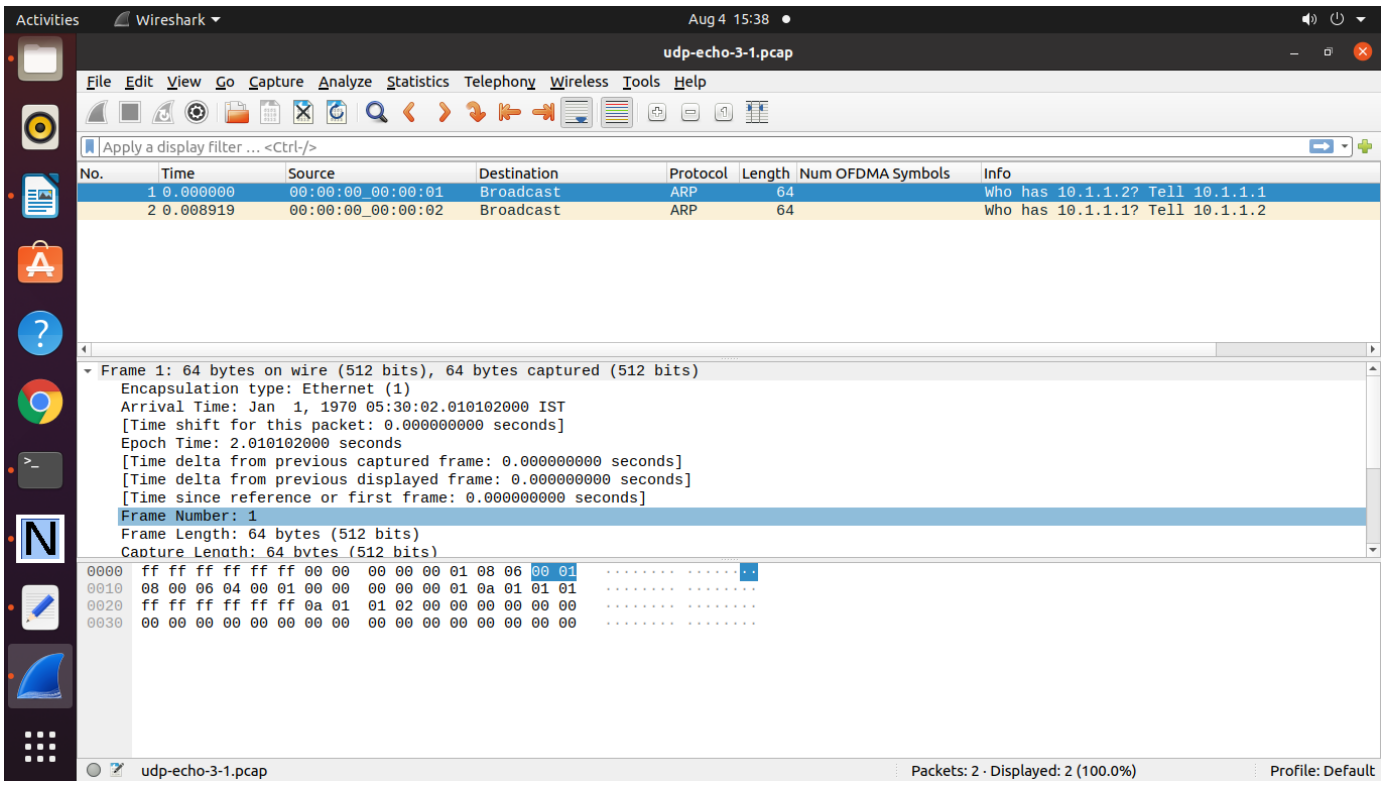

Activities
Terminal
Aug 4 15:35
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
UdpEchoServerApplication:HandleRead(0x559134d250d0, 0x559134e6b260)
At time +2.01592s server received 1024 bytes from 10.1.1.1 port 49153
Echoing packet
At time +2.01592s server sent 1024 bytes to 10.1.1.1 port 49153
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
UdpEchoClientApplication:HandleRead(0x559134d9e0a0, 0x559134e6b930)
At time +2.02484s client received 1024 bytes from 10.1.1.2 port 9
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
UdpEchoClientApplication:StopApplication(0x559134d9e0a0)
UdpEchoServerApplication:StopApplication(0x559134d250d0)
UdpEchoClientApplication:DoDispose(0x559134d9e0a0)
UdpEchoServerApplication:DoDispose(0x559134d250d0)
UdpEchoServerApplication:~UdpEchoServer(0x559134d250d0)
Done.
UdpEchoClientApplication:~UdpEchoClient(0x559134d9e0a0)
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$

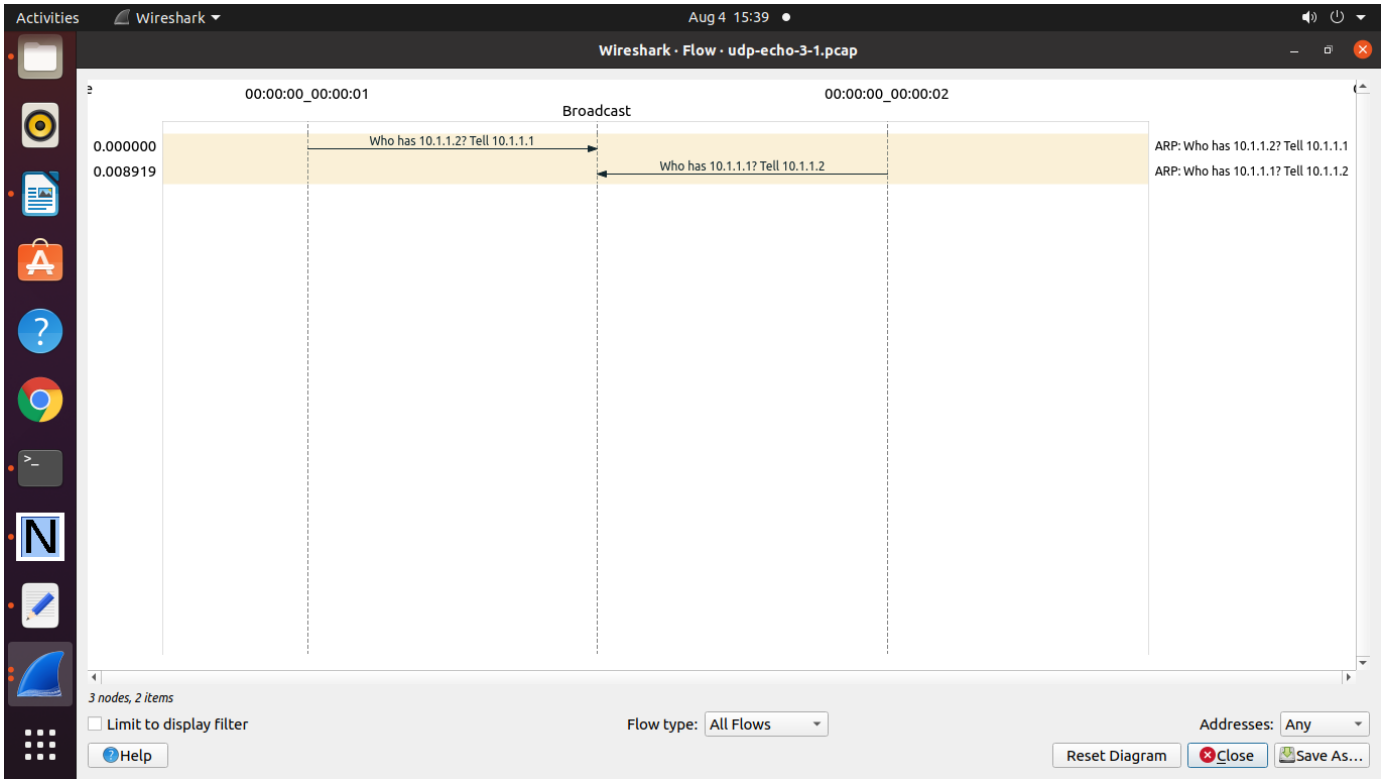

Activities
Python
Aug 4 15:35
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
python
0
10
20
30
40
50
60
Zoom: 1.995
Speed: 1.000
Time: 2.700000 s
Snapshot
Simulate (F3)
Advanced
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary


Activities
NetAnim
Aug 4 15:36
Cancel
Searching in ns-3.32
Open
udp
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
Name
Location
Size
Type
Modified
udp.xml 4.5 kB Markup 15:36
udp-cs.xml 150.4 kB Markup 15:21
udp-cs-0-1.pcap 173.9 kB Packet Capture (PCAP) 15:21
udp-cs-1-1.pcap 173.9 kB Packet Capture (PCAP) 15:21
udpcs.txt 834.9 kB Text 15:27
udp-echo.tr 7.6 kB Text 15:35
udp-echo-0-1.pcap 2.5 kB Packet Capture (PCAP) 15:35
udp-echo-1-1.pcap 2.5 kB Packet Capture (PCAP) 15:35
udp-echo-2-1.pcap 184 bytes Packet Capture (PCAP) 15:35
udp-echo-3-1.pcap 184 bytes Packet Capture (PCAP) 15:35
All Files


Activities
NetAnim
Aug 4 15:37
NetAnim
Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
Lines 5
Node Size 5
IP
MAC
T
0.0,0.0
49.5,0.0
99.0,0.0
0.0,44.0
49.5,44.0
99.0,44.0
0.0,88.0
49.5,88.0
99.0,88.0
0
1
2
3
Playing


Activities
Terminal
Aug 4 15:38
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ wireshark udp-echo-3-1.pcap


Activities
Wireshark
Aug 4 15:38
udp-echo-3-1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Num OFDMA Symbols Info
1 0.000000 00:00:00_00:00:01 Broadcast ARP 64 Who has 10.1.1.2? Tell 10.1.1.1
2 0.008919 00:00:00_00:00:02 Broadcast ARP 64 Who has 10.1.1.1? Tell 10.1.1.2
Frame 1: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)
Encapsulation type: Ethernet (1)
Arrival Time: Jan 1, 1970 05:30:02.010102000 IST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 2.010102000 seconds
[Time delta from previous captured frame: 0.000000000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
Frame Length: 64 bytes (512 bits)
Capture Length: 64 bytes (512 bits)
0000 ff ff ff ff ff ff 00 00 00 00 00 01 08 06 00 01
0010 08 00 06 04 00 01 00 00 00 00 00 01 0a 01 01 01
0020 ff ff ff ff ff ff 0a 01 01 02 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
udp-echo-3-1.pcap
Packets: 2 · Displayed: 2 (100.0%)
Profile: Default


Activities
Wireshark
Aug 4 15:39
Wireshark · Flow · udp-echo-3-1.pcap
00:00:00_00:00:01
00:00:00_00:00:02
Broadcast
0.000000
Who has 10.1.1.2? Tell 10.1.1.1
ARP: Who has 10.1.1.2? Tell 10.1.1.1
0.008919
Who has 10.1.1.1? Tell 10.1.1.2
ARP: Who has 10.1.1.1? Tell 10.1.1.2
3 nodes, 2 items
Limit to display filter
Flow type: All Flows
Addresses: Any
Help
Reset Diagram
Close
Save As...

==========

Practical 13: Program to Simulate Wireless Wi-Fi Network .

```
/* -*- 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
 *
 *  (Adapted from third.cc)
 */

#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/csma-module.h"
```

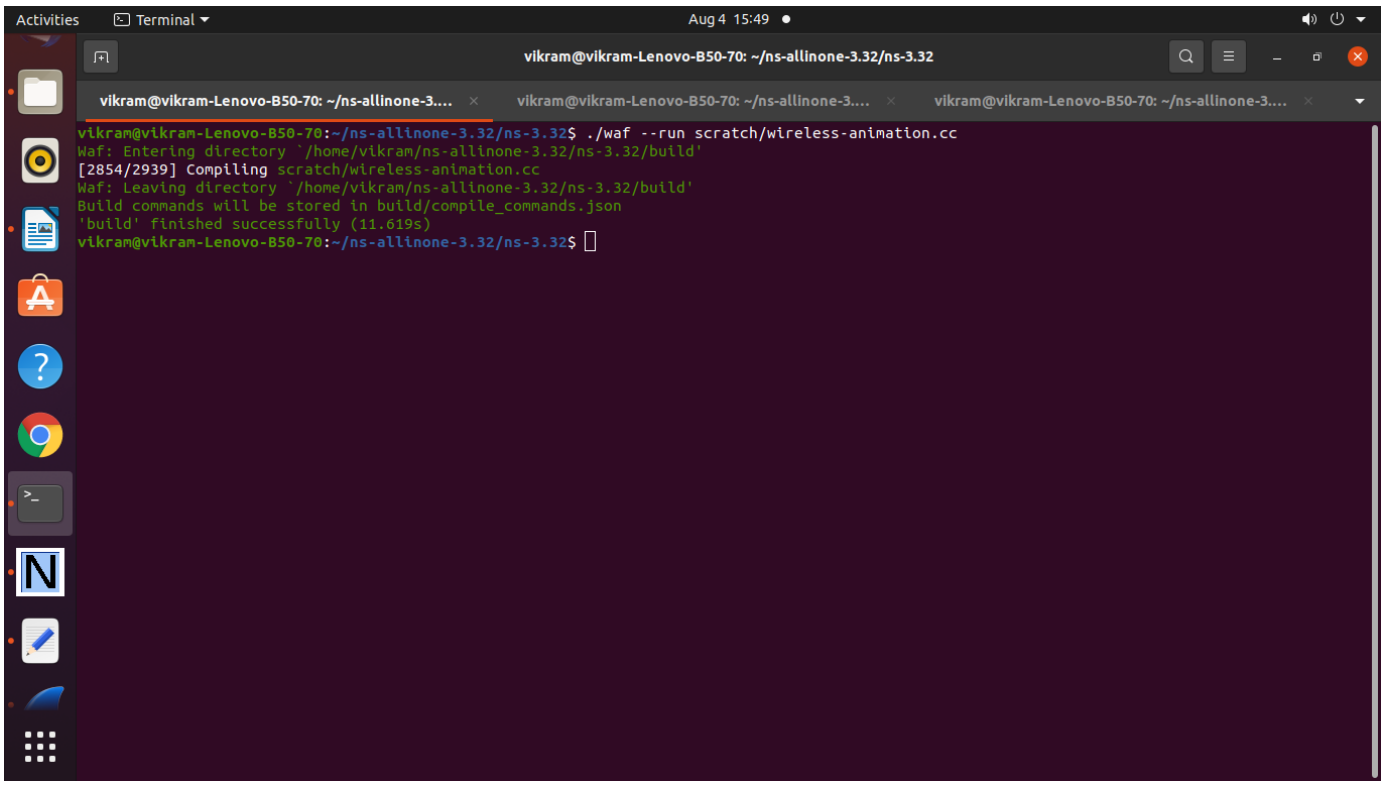
```
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/internet-module.h"
#include "ns3/netanim-module.h"
#include "ns3/basic-energy-source.h"
#include "ns3/simple-device-energy-model.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/ssid.h"
#include "ns3/wifi-radio-energy-model.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("WirelessAnimationExample");

int
main (int argc, char *argv[])
{
 uint32_t nWifi = 20;
 CommandLine cmd (__FILE__);
 cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);

 cmd.Parse (argc,argv);
 NodeContainer allNodes;
 NodeContainer wifiStaNodes;
 wifiStaNodes.Create (nWifi);
 allNodes.Add (wifiStaNodes);
 NodeContainer wifiApNode ;
 wifiApNode.Create (1);
 allNodes.Add (wifiApNode);

 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");
 mac.SetType ("ns3::StaWifiMac",
              "Ssid", SsidValue (ssid),
              "ActiveProbing", BooleanValue (false));

 NetDeviceContainer staDevices;
```

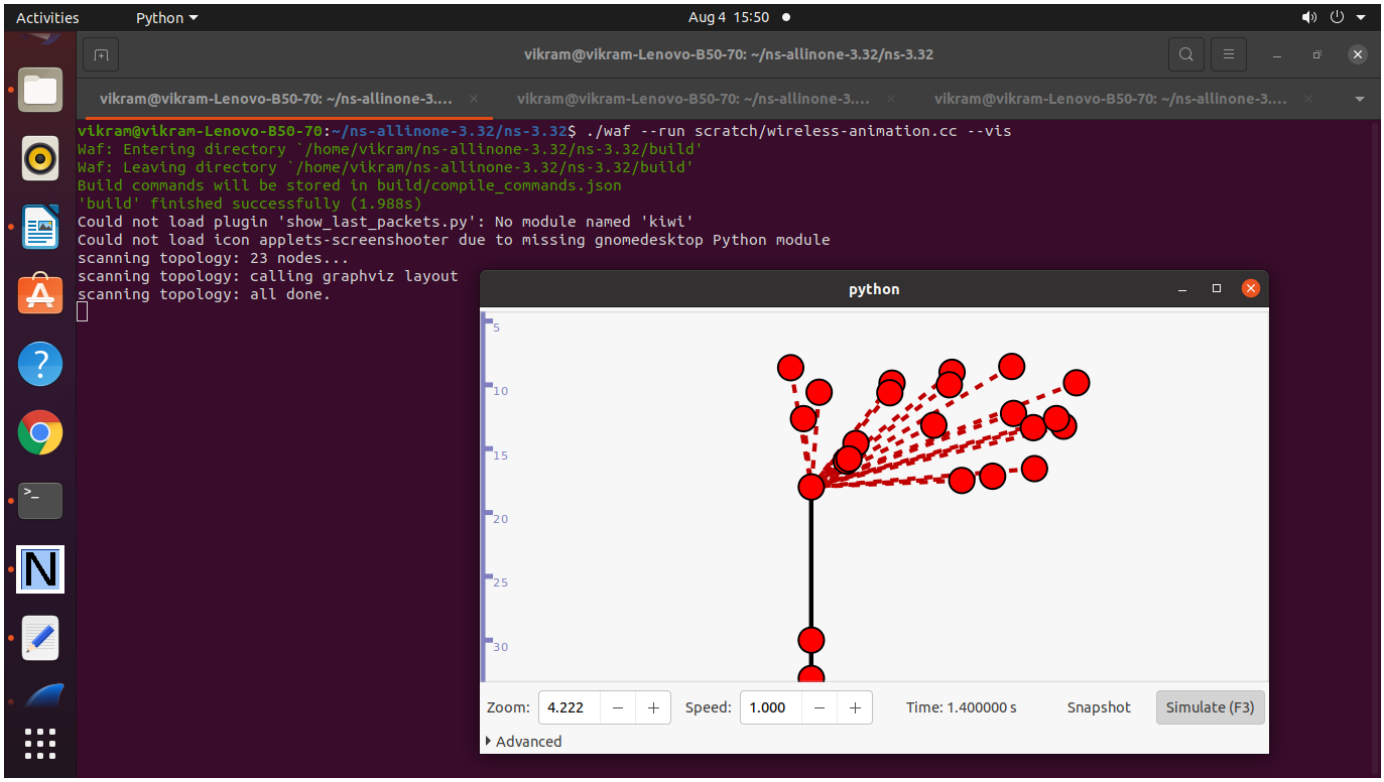
```
staDevices = wifi.Install (phy, mac, wifiStaNodes);
mac.SetType ("ns3::ApWifiMac",
          "Ssid", SsidValue (ssid));

NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);


NodeContainer p2pNodes;
p2pNodes.Add (wifiApNode);
p2pNodes.Create (1);
allNodes.Add (p2pNodes.Get (1));

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

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

NodeContainer csmaNodes;
csmaNodes.Add (p2pNodes.Get (1));
csmaNodes.Create (1);
allNodes.Add (csmaNodes.Get (1));

CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);

// Mobility

MobilityHelper mobility;
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                               "MinX", DoubleValue (10.0),
                               "MinY", DoubleValue (10.0),
                               "DeltaX", DoubleValue (5.0),
                               "DeltaY", DoubleValue (2.0),
                               "GridWidth", UintegerValue (5),
                               "LayoutType", StringValue ("RowFirst"));
mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                           "Bounds", RectangleValue (Rectangle (-50, 50, -25, 50)));
mobility.Install (wifiStaNodes);
```

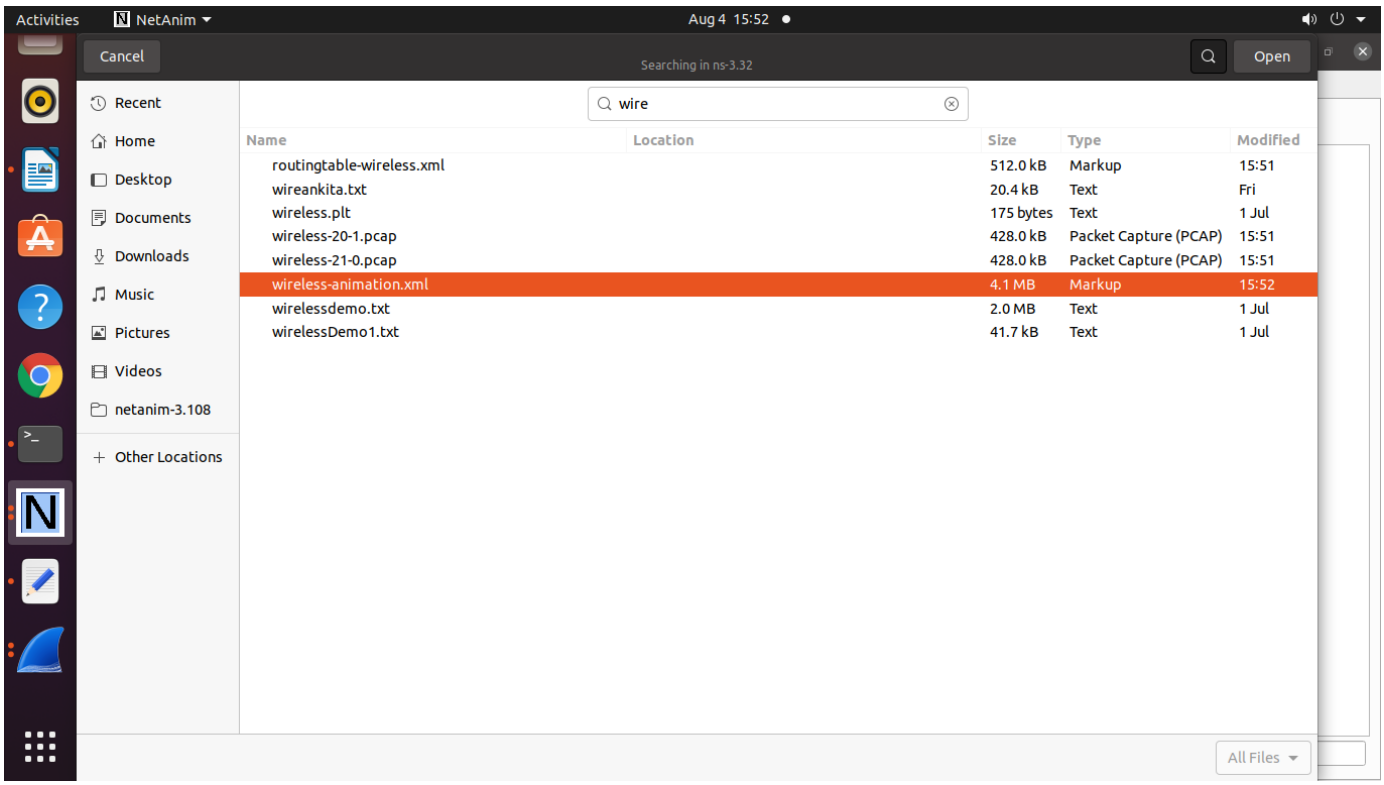
```
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility.Install (wifiApNode);
  AnimationInterface::SetConstantPosition (p2pNodes.Get (1), 10, 30);
  AnimationInterface::SetConstantPosition (csmaNodes.Get (1), 10, 33);

  Ptr<BasicEnergySource> energySource = CreateObject<BasicEnergySource>();
  Ptr<WifiRadioEnergyModel> energyModel =
CreateObject<WifiRadioEnergyModel>();

  energySource->SetInitialEnergy (300);
  energyModel->SetEnergySource (energySource);
  energySource->AppendDeviceEnergyModel (energyModel);

  // aggregate energy source to node
  wifiApNode.Get (0)->AggregateObject (energySource);

  // Install internet stack

  InternetStackHelper stack;
  stack.Install (allNodes);

  // Install Ipv4 addresses

  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);
  address.SetBase ("10.1.3.0", "255.255.255.0");
  Ipv4InterfaceContainer staInterfaces;
  staInterfaces = address.Assign (staDevices);
  Ipv4InterfaceContainer apInterface;
  apInterface = address.Assign (apDevices);

  // Install applications

  UdpEchoServerHelper echoServer (9);
  ApplicationContainer serverApps = echoServer.Install (csmaNodes.Get (1));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (15.0));
  UdpEchoClientHelper echoClient (csmaInterfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (10));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.)));
```

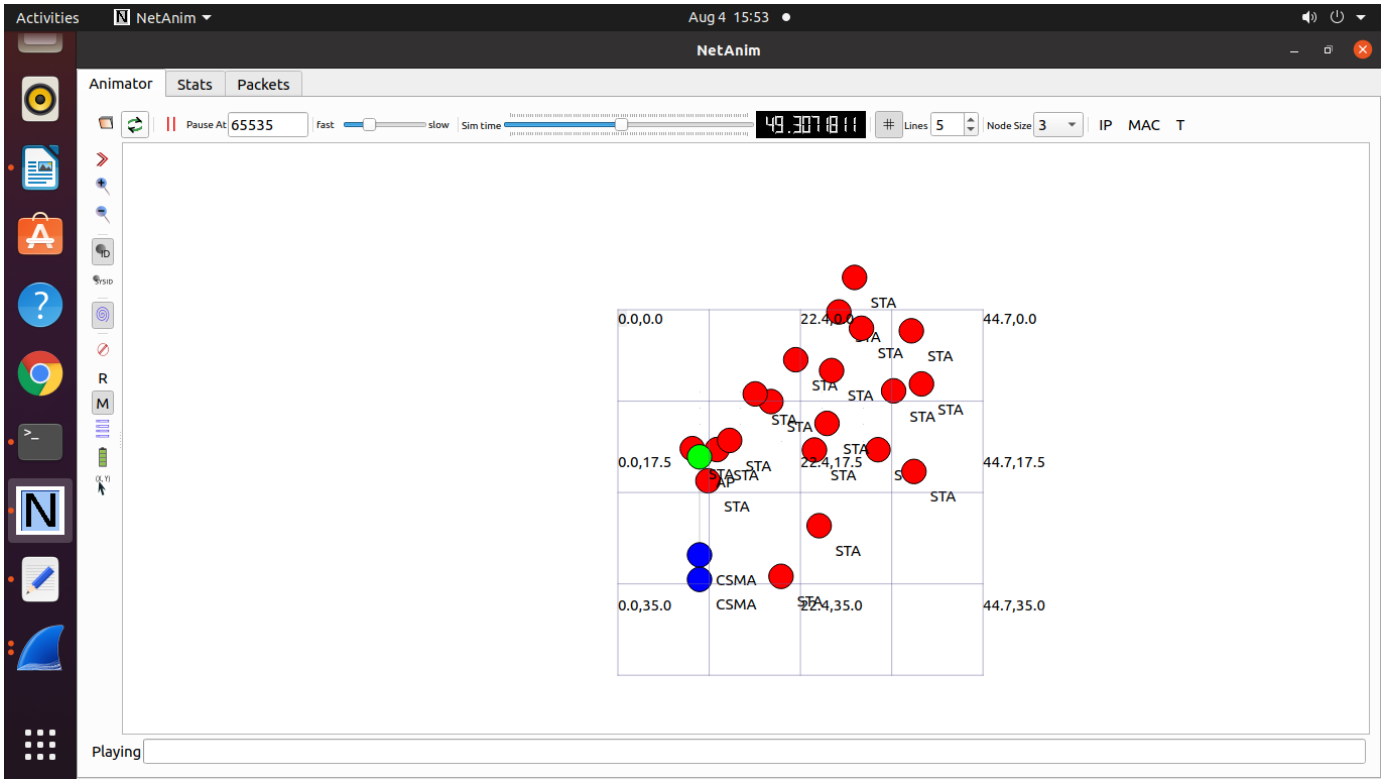
```
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer clientApps = echoClient.Install (wifiStaNodes);
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (15.0));

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
  Simulator::Stop (Seconds (15.0));

  AnimationInterface anim ("wireless-animation.xml"); // Mandatory
  for (uint32_t i = 0; i < wifiStaNodes.GetN (); ++i)
    {
      anim.UpdateNodeDescription (wifiStaNodes.Get (i), "STA"); // Optional
      anim.UpdateNodeColor (wifiStaNodes.Get (i), 255, 0, 0); // Optional
    }
  for (uint32_t i = 0; i < wifiApNode.GetN (); ++i)
    {
      anim.UpdateNodeDescription (wifiApNode.Get (i), "AP"); // Optional
      anim.UpdateNodeColor (wifiApNode.Get (i), 0, 255, 0); // Optional
    }
  for (uint32_t i = 0; i < csmaNodes.GetN (); ++i)
    {
      anim.UpdateNodeDescription (csmaNodes.Get (i), "CSMA"); // Optional
      anim.UpdateNodeColor (csmaNodes.Get (i), 0, 0, 255); // Optional
    }

  anim.EnablePacketMetadata (true); // Optional
  anim.EnableIpv4RouteTracking ("routingtable-wireless.xml", Seconds (0),
Seconds (5), Seconds (0.25)); //Optional
  anim.EnableWifiMacCounters (Seconds (0), Seconds (10)); //Optional
  anim.EnableWifiPhyCounters (Seconds (0), Seconds (10)); //Optional
  pointToPoint.EnablePcapAll("wireless");
  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

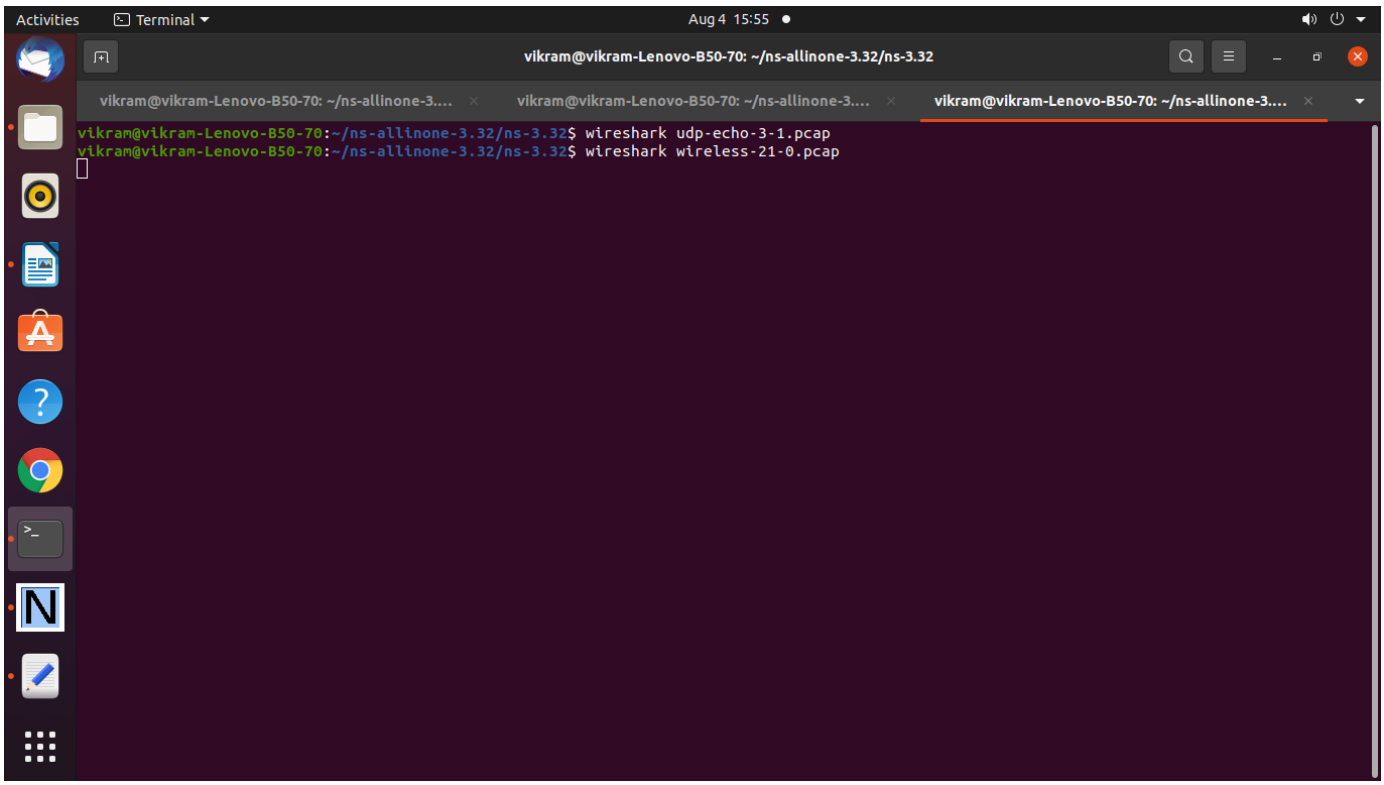
Output :

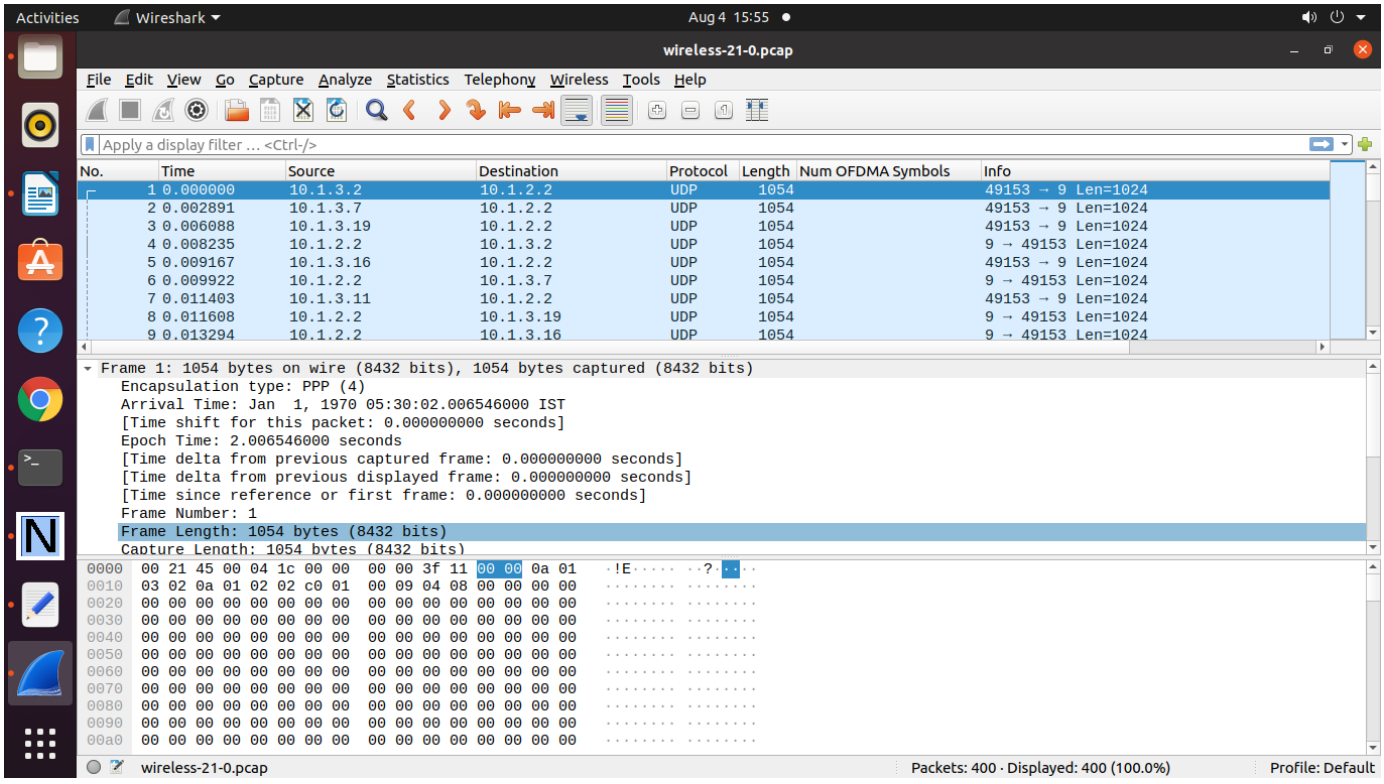

Activities
Terminal
Aug 4 15:49
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/wireless-animation.cc
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
[2854/2939] Compiling scratch/wireless-animation.cc
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (11.619s)
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$

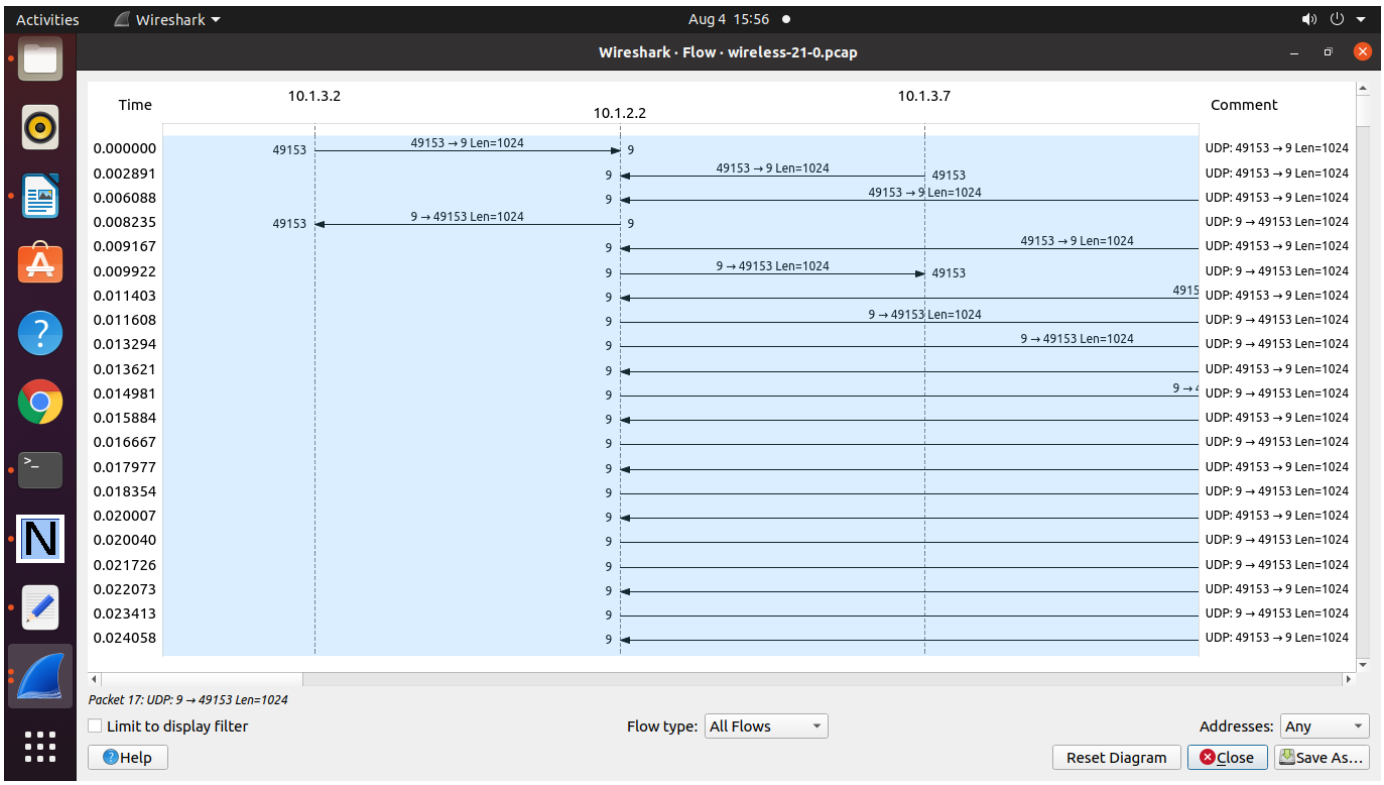

Activities
Python
Aug 4 15:50
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/wireless-animation.cc --vis
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.988s)
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: 23 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
python
5
10
15
20
25
30
Zoom: 4.222
Speed: 1.000
Time: 1.400000 s
Snapshot
Simulate (F3)
Advanced


Activities
NetAnim
Aug 4 15:52
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
wire
Name
Location
Size
Type
Modified
routingtable-wireless.xml
512.0 kB
Markup
15:51
wireankita.txt
20.4 kB
Text
Fri
wireless.plt
175 bytes
Text
1 Jul
wireless-20-1.pcap
428.0 kB
Packet Capture (PCAP)
15:51
wireless-21-0.pcap
428.0 kB
Packet Capture (PCAP)
15:51
wireless-animation.xml
4.1 MB
Markup
15:52
wirelessdemo.txt
2.0 MB
Text
1 Jul
wirelessDemo1.txt
41.7 kB
Text
1 Jul
All Files


Activities
NetAnim
Aug 4 15:53
NetAnim
Animator
Stats
Packets
Pause At
65535
fast
slow
Sim time
Lines
5
Node Size
3
IP
MAC
T
0.0,0.0
22.4,0.0
44.7,0.0
0.0,17.5
22.4,17.5
44.7,17.5
0.0,35.0
22.4,35.0
44.7,35.0
STA
AP
CSMA
Playing


Activities
Terminal
Aug 4 15:55
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ wireshark udp-echo-3-1.pcap
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ wireshark wireless-21-0.pcap


Activities
Wireshark
Aug 4 15:55
wireless-21-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Num OFDMA Symbols Info
1 0.000000 10.1.3.2 10.1.2.2 UDP 1054 49153 → 9 Len=1024
2 0.002891 10.1.3.7 10.1.2.2 UDP 1054 49153 → 9 Len=1024
3 0.006088 10.1.3.19 10.1.2.2 UDP 1054 49153 → 9 Len=1024
4 0.008235 10.1.2.2 10.1.3.2 UDP 1054 9 → 49153 Len=1024
5 0.009167 10.1.3.16 10.1.2.2 UDP 1054 49153 → 9 Len=1024
6 0.009922 10.1.2.2 10.1.3.7 UDP 1054 9 → 49153 Len=1024
7 0.011403 10.1.3.11 10.1.2.2 UDP 1054 49153 → 9 Len=1024
8 0.011608 10.1.2.2 10.1.3.19 UDP 1054 9 → 49153 Len=1024
9 0.013294 10.1.2.2 10.1.3.16 UDP 1054 9 → 49153 Len=1024
Frame 1: 1054 bytes on wire (8432 bits), 1054 bytes captured (8432 bits)
Encapsulation type: PPP (4)
Arrival Time: Jan 1, 1970 05:30:02.006546000 IST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 2.006546000 seconds
[Time delta from previous captured frame: 0.000000000 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 0.000000000 seconds]
Frame Number: 1
Frame Length: 1054 bytes (8432 bits)
Capture Length: 1054 bytes (8432 bits)
0000 00 21 45 00 04 1c 00 00 00 00 3f 11 00 00 0a 01
0010 03 02 0a 01 02 02 c0 01 00 09 04 08 00 00 00 00
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
wireless-21-0.pcap
Packets: 400 · Displayed: 400 (100.0%)
Profile: Default



Practical 13 : Program to simulate WiMax wireless network.

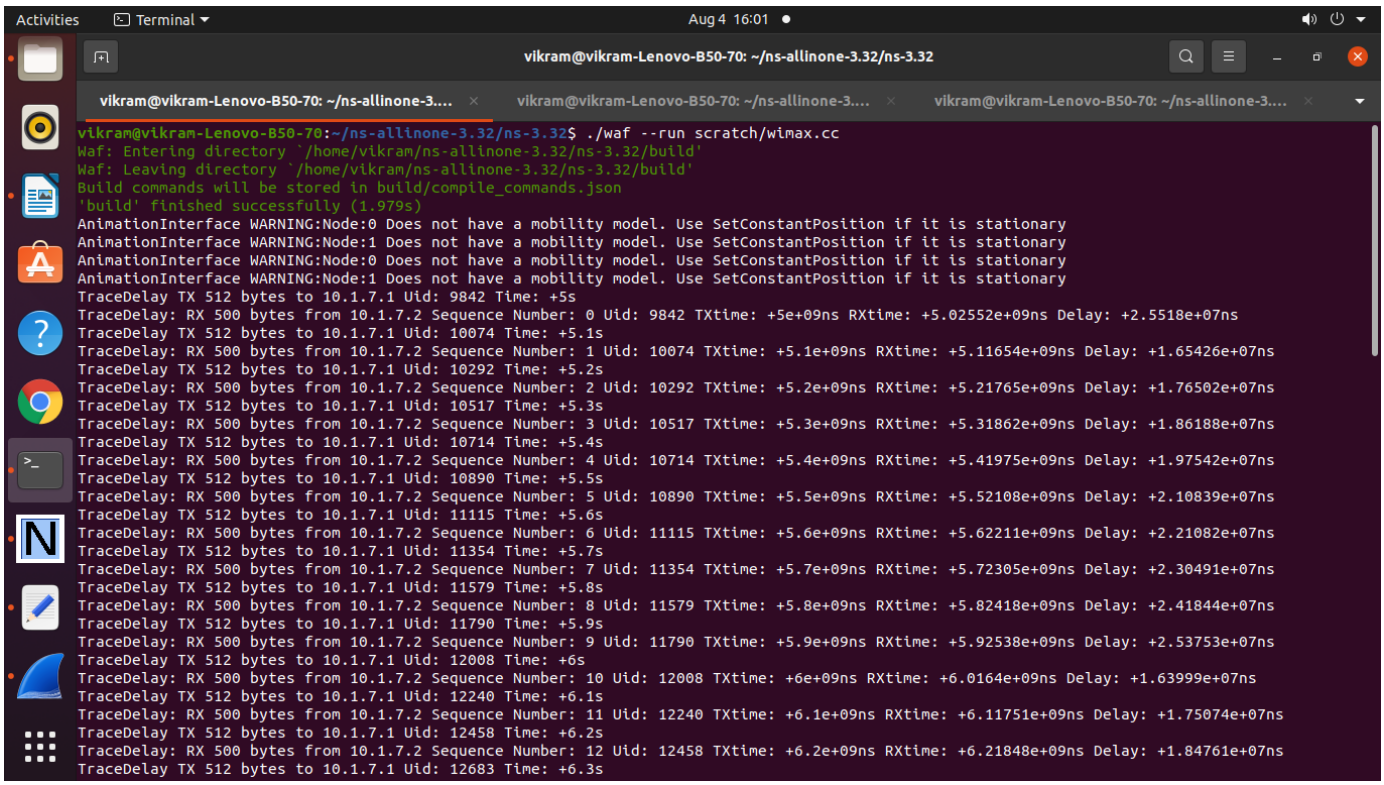
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/config-store-module.h"
#include "ns3/wimax-module.h"
#include "ns3/internet-module.h"
#include "ns3/global-route-manager.h"
#include "ns3/ipcs-classifier-record.h"
#include "ns3/service-flow.h"
#include <iostream>
#include "ns3/netanim-module.h"
#include "ns3/net-device-container.h"
#include "ns3/netanim-module.h"

using namespace ns3;
```

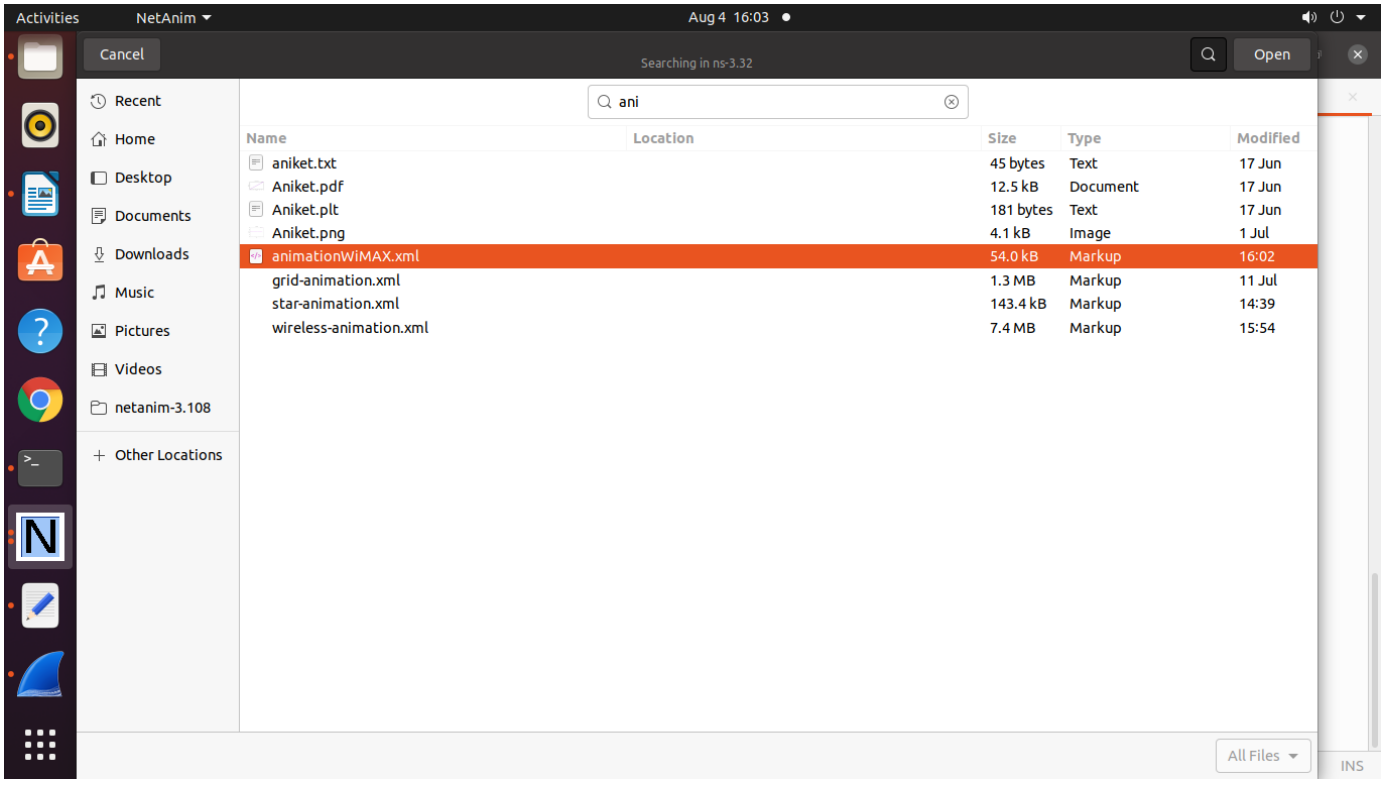
```
int main (int argc, char *argv[]){

WimaxHelper::SchedulerType scheduler
=WimaxHelper::SCHED_TYPE_SIMPLE;

LogComponentEnable ("UdpClient", LOG_LEVEL_INFO);
LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);

NodeContainer ssNodes;
NodeContainer bsNodes;

ssNodes.Create (2);
bsNodes.Create (1);

WimaxHelper wimax;
NetDeviceContainer ssDevs;
NetDeviceContainer bsDevs;

ssDevs = wimax.Install(ssNodes,
WimaxHelper::DEVICE_TYPE_SUBSCRIBER_STATION,
WimaxHelper::SIMPLE_PHY_TYPE_OFDM, scheduler);

bsDevs = wimax.Install(bsNodes,
WimaxHelper::DEVICE_TYPE_BASE_STATION,WimaxHelper::SIMPLE_PHY_T
YPE_OFDM,
scheduler);

Ptr<SubscriberStationNetDevice> ss[2];

for (int i = 0; i < 2; i++) {
ss[i] = ssDevs.Get (i)->
GetObject<SubscriberStationNetDevice> ();

ss[i]->SetModulationType
(WimaxPhy::MODULATION_TYPE_QAM64_34);
}


Ptr<BaseStationNetDevice> bs;
bs = bsDevs.Get (0)->
GetObject<BaseStationNetDevice> ();

InternetStackHelper stack;
stack.Install (bsNodes);
stack.Install (ssNodes);
```

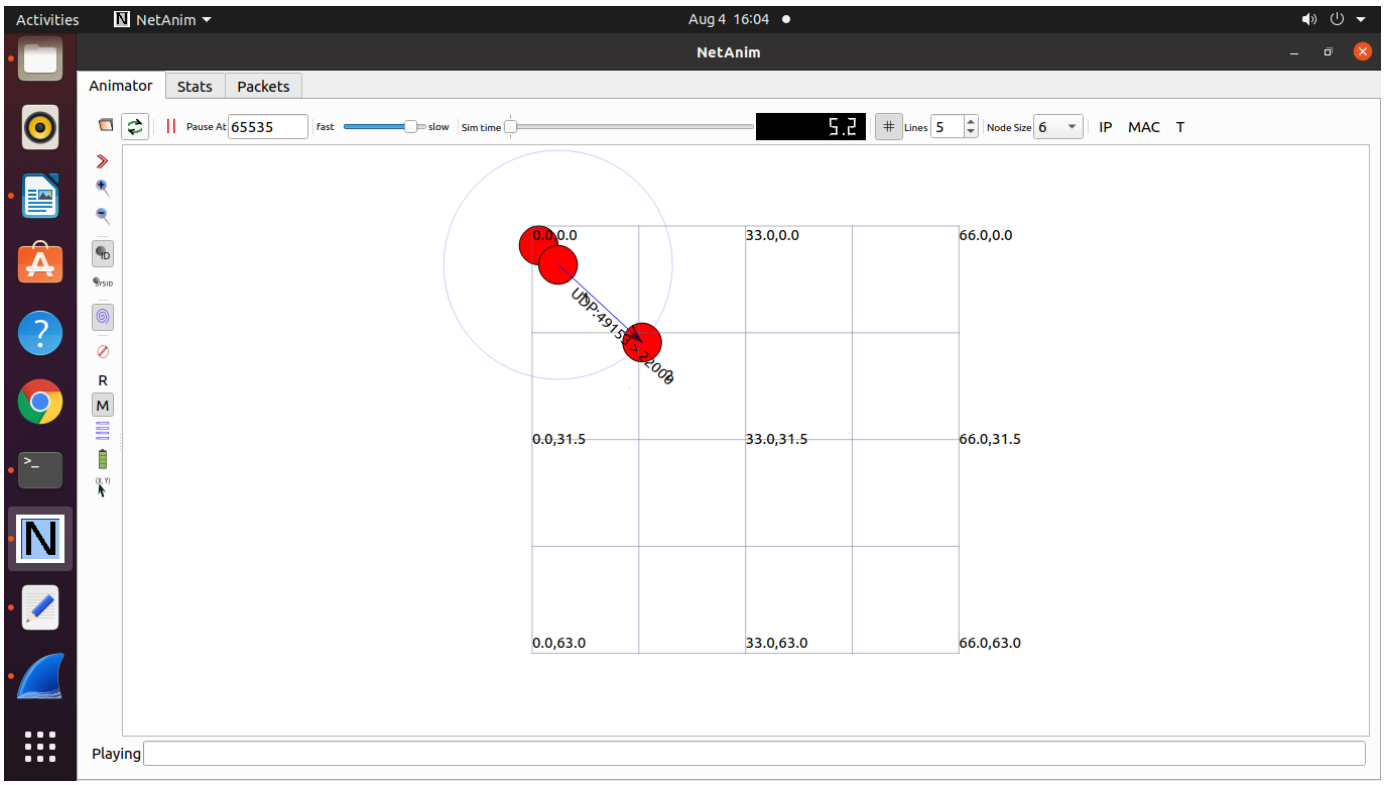
```
Ipv4AddressHelper address;
address.SetBase ("10.1.7.0", "255.255.255.0");

Ipv4InterfaceContainer SSinterfaces = address.Assign (ssDevs);
Ipv4InterfaceContainer BSinterface = address.Assign (bsDevs);

Ptr<ListPositionAllocator>
positionAlloc = CreateObject
<ListPositionAllocator> ();

positionAlloc->Add (Vector(15, 25, 0));
positionAlloc->Add (Vector(25, 15, 0));
positionAlloc->Add (Vector(5, 15, 0));

MobilityHelper bs_mobility;
bs_mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
bs_mobility.SetPositionAllocator(positionAlloc);
bs_mobility.Install(bsNodes);

UdpServerHelper udpServer;
ApplicationContainer serverApps;
UdpClientHelper udpClient;
ApplicationContainer clientApps;

udpServer = UdpServerHelper (22000);

serverApps = udpServer.Install (ssNodes.Get (0));
serverApps.Start (Seconds (4.0));
serverApps.Stop (Seconds (10.0));

udpClient=UdpClientHelper(SSinterfaces.GetAddress (0),22000);
udpClient.SetAttribute ("MaxPackets", UintegerValue (15000));
udpClient.SetAttribute ("Interval", TimeValue (Seconds (0.1)));
udpClient.SetAttribute ("PacketSize", UintegerValue (512));
clientApps = udpClient.Install (ssNodes.Get (1));
clientApps.Start (Seconds (5.0));
clientApps.Stop (Seconds (9.5));
Simulator::Stop (Seconds (15.0));

IpcsClassifierRecord DlClassifierUgs (Ipv4Address ("0.0.0.0"),
Ipv4Mask ("0.0.0.0"), SSinterfaces.GetAddress (0), Ipv4Mask ("255.255.255.255"),
0, 65000,
22000, 22000, 17, 1);
```

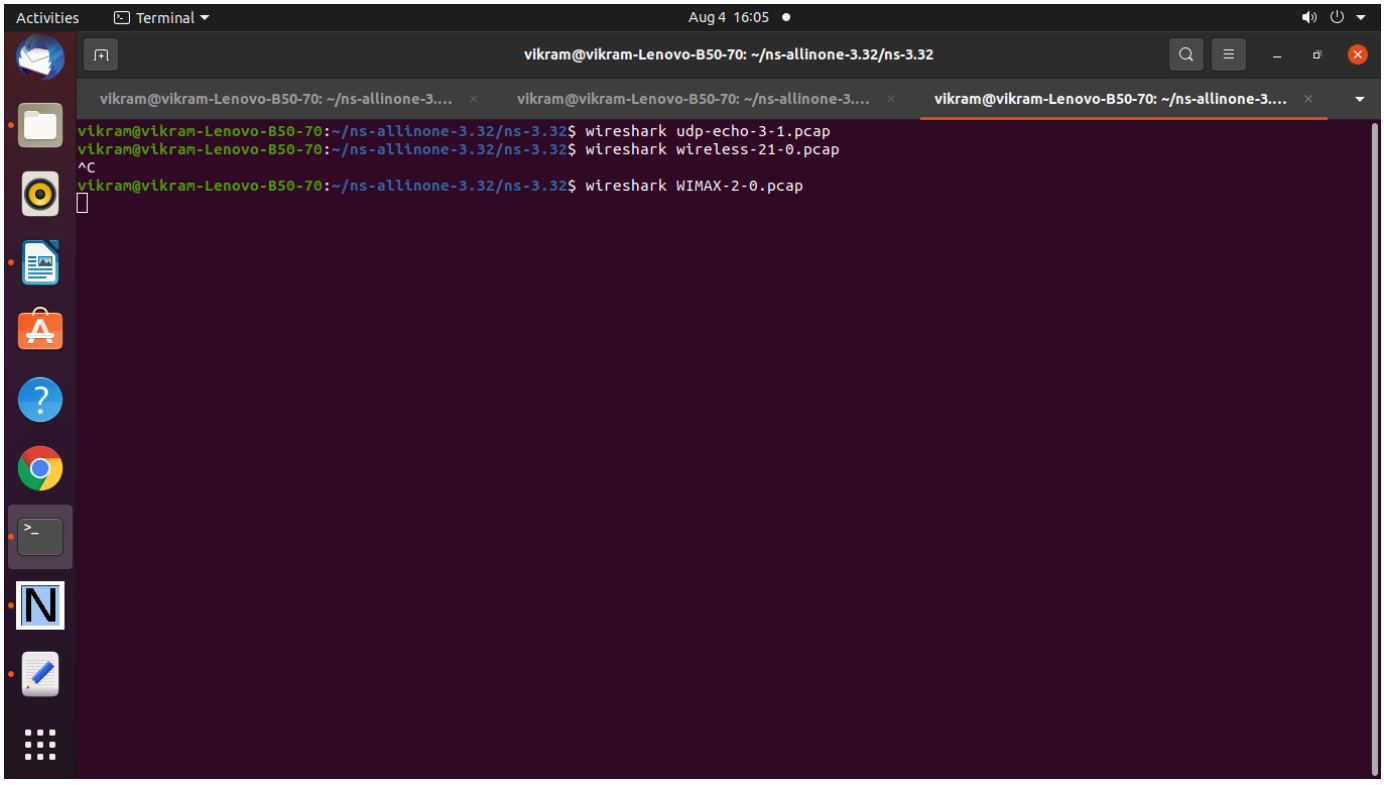
```
ServiceFlow DlServiceFlowUgs = wimax.CreateServiceFlow
(ServiceFlow::SF_DIRECTION_DOWN,ServiceFlow::SF_TYPE_RTPS,
DlClassifierUgs);
ss[0]->AddServiceFlow (DlServiceFlowUgs);

IpcsClassifierRecord UlClassifierUgs (SSinterfaces.GetAddress(1), Ipv4Mask
("255.255.255.255"),
Ipv4Address ("0.0.0.0"),Ipv4Mask ("0.0.0.0"), 0, 65000, 22000, 22000, 17, 1);
ServiceFlow UlServiceFlowUgs =
wimax.CreateServiceFlow(ServiceFlow::SF_DIRECTION_UP,
ServiceFlow::SF_TYPE_RTPS,UlClassifierUgs);
ss[1]->AddServiceFlow (UlServiceFlowUgs);

AnimationInterface anim("animationWiMAX.xml");
anim.SetConstantPosition (ssNodes.Get(0),1.0,3.0);
anim.SetConstantPosition (ssNodes.Get(1),4.0,6.0);
anim.SetConstantPosition (bsNodes.Get(0),17.0,18.0);
anim.EnablePacketMetadata(true);
wimax.EnablePcapAll("WIMAX");

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

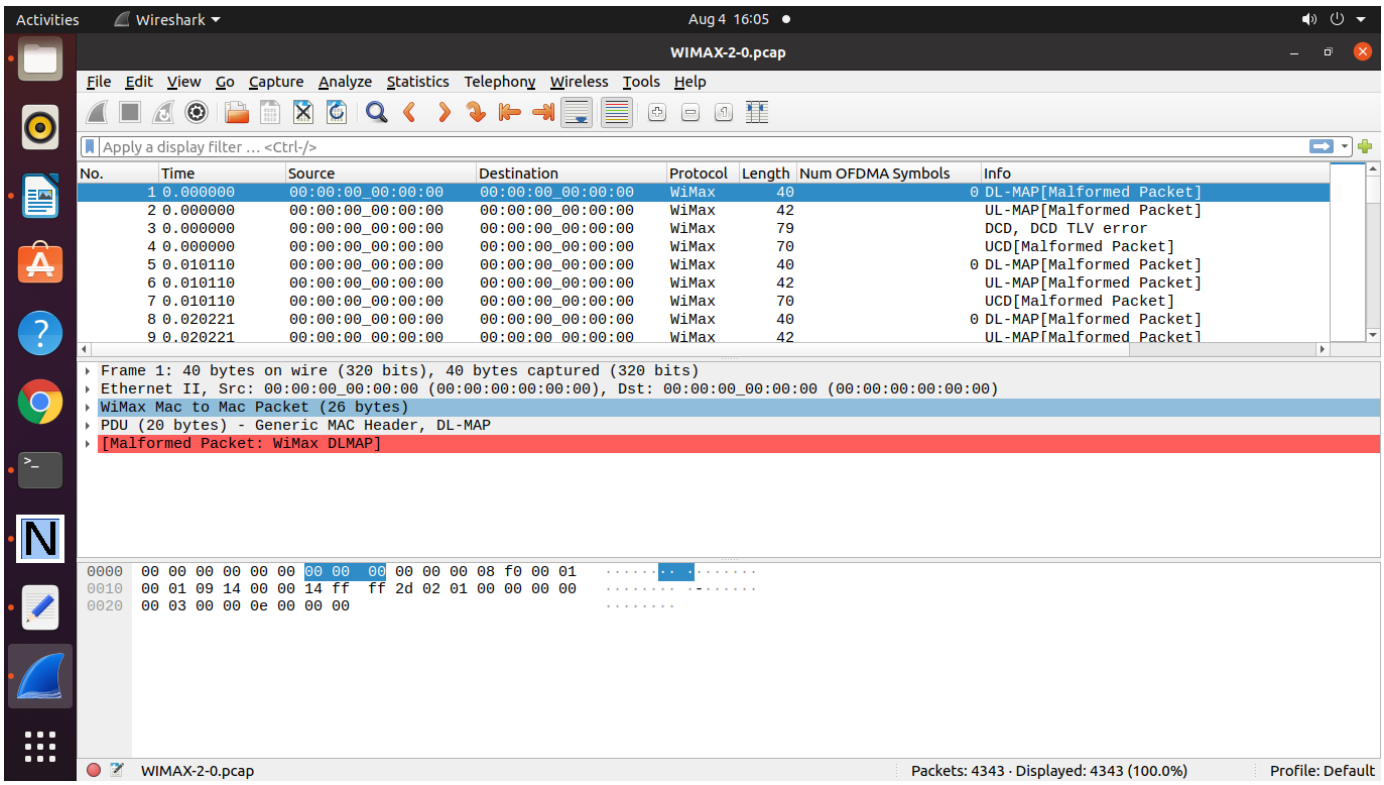
output :

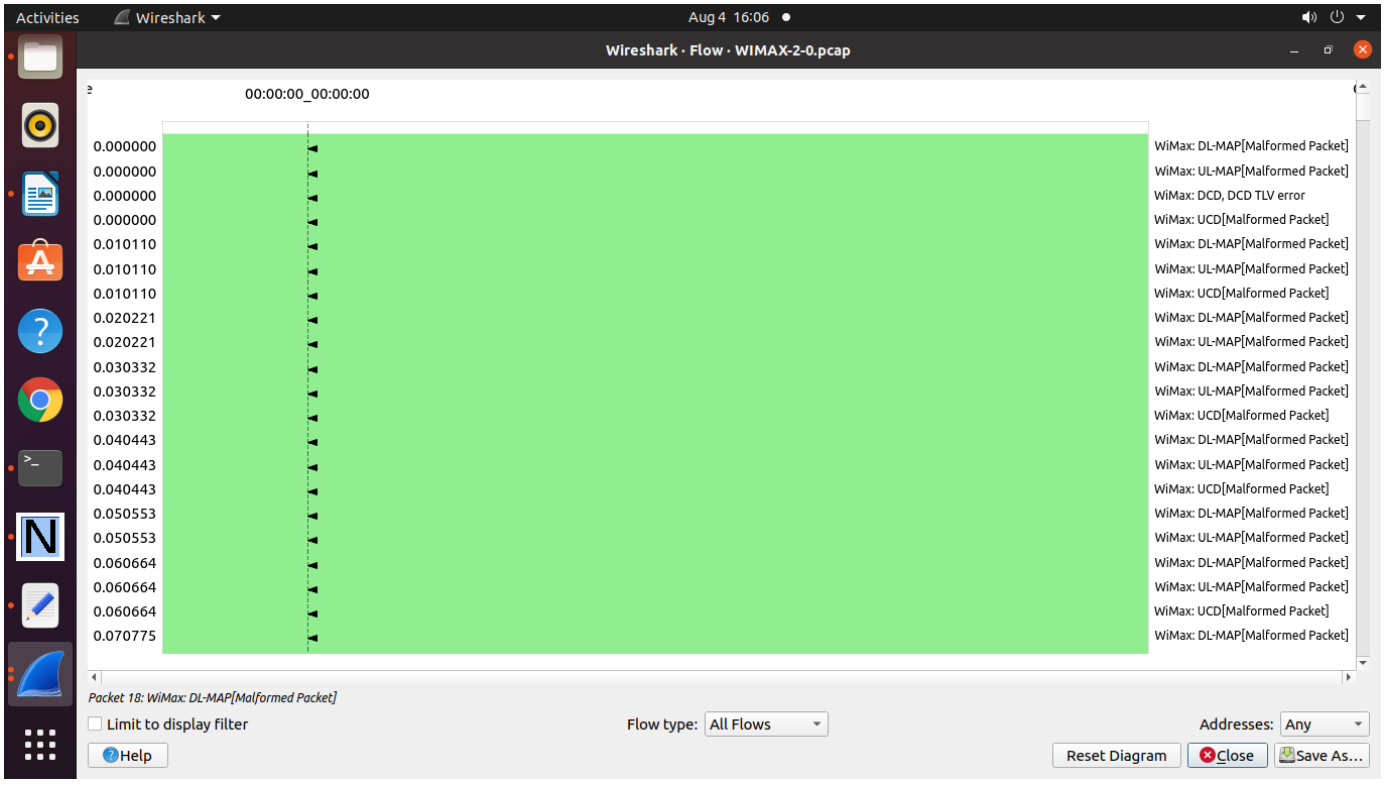

Activities
Terminal
Aug 4 16:01
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/wimax.cc
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.979s)
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 9842 Time: +5s
TraceDelay: RX 500 bytes from 10.1.7.2 Sequence Number: 0 Uid: 9842 TXtime: +5e+09ns RXtime: +5.02552e+09ns Delay: +2.5518e+07ns
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 10074 Time: +5.1s
TraceDelay: RX 500 bytes from 10.1.7.2 Sequence Number: 1 Uid: 10074 TXtime: +5.1e+09ns RXtime: +5.11654e+09ns Delay: +1.65426e+07ns
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 10292 Time: +5.2s
TraceDelay: RX 500 bytes from 10.1.7.2 Sequence Number: 2 Uid: 10292 TXtime: +5.2e+09ns RXtime: +5.21765e+09ns Delay: +1.76502e+07ns
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 10517 Time: +5.3s
TraceDelay: RX 500 bytes from 10.1.7.2 Sequence Number: 3 Uid: 10517 TXtime: +5.3e+09ns RXtime: +5.31862e+09ns Delay: +1.86188e+07ns
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 10714 Time: +5.4s
TraceDelay: RX 500 bytes from 10.1.7.2 Sequence Number: 4 Uid: 10714 TXtime: +5.4e+09ns RXtime: +5.41975e+09ns Delay: +1.97542e+07ns
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 10890 Time: +5.5s
TraceDelay: RX 500 bytes from 10.1.7.2 Sequence Number: 5 Uid: 10890 TXtime: +5.5e+09ns RXtime: +5.52108e+09ns Delay: +2.10839e+07ns
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 11115 Time: +5.6s
TraceDelay: RX 500 bytes from 10.1.7.2 Sequence Number: 6 Uid: 11115 TXtime: +5.6e+09ns RXtime: +5.62211e+09ns Delay: +2.21082e+07ns
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 11354 Time: +5.7s
TraceDelay: RX 500 bytes from 10.1.7.2 Sequence Number: 7 Uid: 11354 TXtime: +5.7e+09ns RXtime: +5.72305e+09ns Delay: +2.30491e+07ns
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 11579 Time: +5.8s
TraceDelay: RX 500 bytes from 10.1.7.2 Sequence Number: 8 Uid: 11579 TXtime: +5.8e+09ns RXtime: +5.82418e+09ns Delay: +2.41844e+07ns
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 11790 Time: +5.9s
TraceDelay: RX 500 bytes from 10.1.7.2 Sequence Number: 9 Uid: 11790 TXtime: +5.9e+09ns RXtime: +5.92538e+09ns Delay: +2.53753e+07ns
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 12008 Time: +6s
TraceDelay: RX 500 bytes from 10.1.7.2 Sequence Number: 10 Uid: 12008 TXtime: +6e+09ns RXtime: +6.0164e+09ns Delay: +1.63999e+07ns
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 12240 Time: +6.1s
TraceDelay: RX 500 bytes from 10.1.7.2 Sequence Number: 11 Uid: 12240 TXtime: +6.1e+09ns RXtime: +6.11751e+09ns Delay: +1.75074e+07ns
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 12458 Time: +6.2s
TraceDelay: RX 500 bytes from 10.1.7.2 Sequence Number: 12 Uid: 12458 TXtime: +6.2e+09ns RXtime: +6.21848e+09ns Delay: +1.84761e+07ns
TraceDelay TX 512 bytes to 10.1.7.1 Uid: 12683 Time: +6.3s


Activities
NetAnim
Aug 4 16:03
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
ani
Name
Location
Size
Type
Modified
aniket.txt
45 bytes
Text
17 Jun
Aniket.pdf
12.5 kB
Document
17 Jun
Aniket.plt
181 bytes
Text
17 Jun
Aniket.png
4.1 kB
Image
1 Jul
animationWiMAX.xml
54.0 kB
Markup
16:02
grid-animation.xml
1.3 MB
Markup
11 Jul
star-animation.xml
143.4 kB
Markup
14:39
wireless-animation.xml
7.4 MB
Markup
15:54
All Files
INS


Activities
NetAnim
Aug 4 16:04
NetAnim
Animator
Stats
Packets
Pause At
65535
fast
slow
Sim time
5.2
Lines
5
Node Size
6
IP
MAC
T
0.0,0.0
33.0,0.0
66.0,0.0
0.0,31.5
33.0,31.5
66.0,31.5
0.0,63.0
33.0,63.0
66.0,63.0
Playing

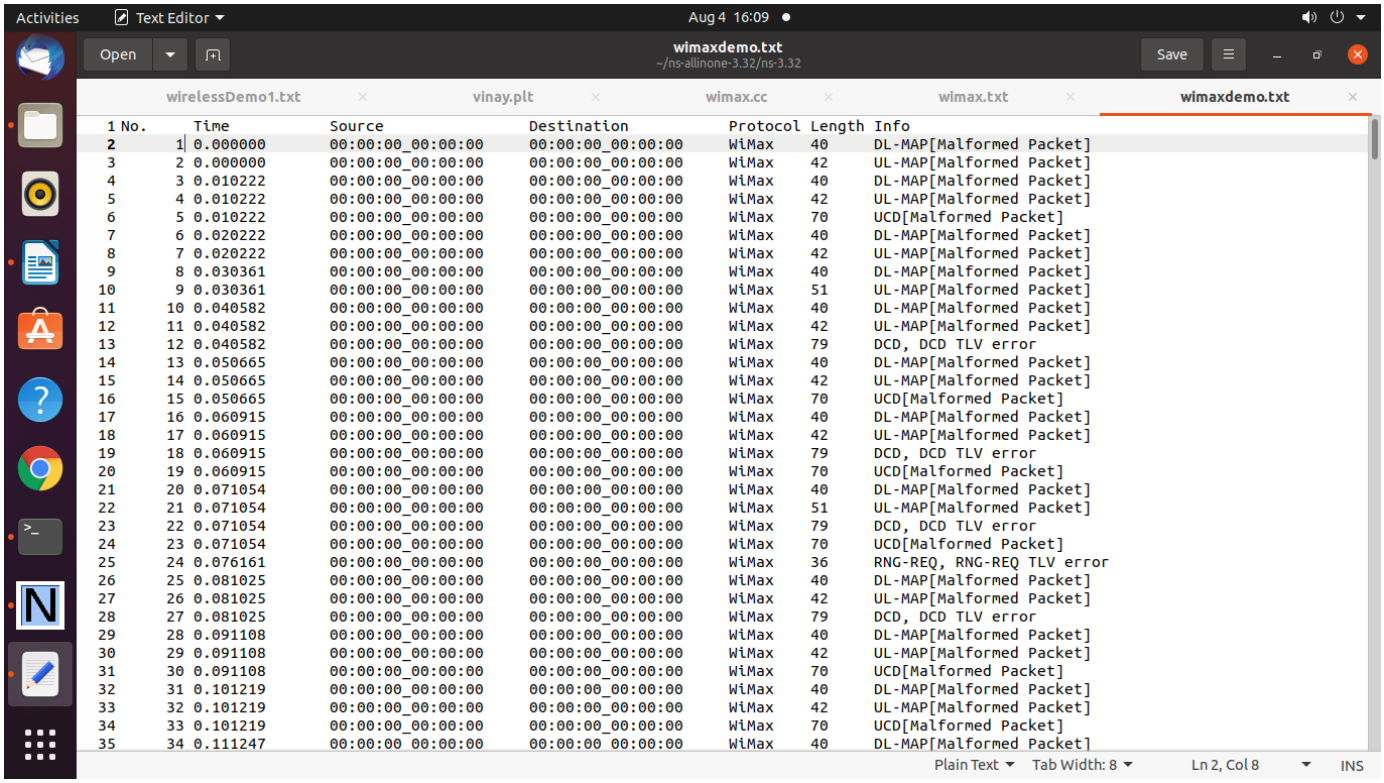

Activities
Terminal
Aug 4 16:05
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ wireshark udp-echo-3-1.pcap
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ wireshark wireless-21-0.pcap
^C
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ wireshark WIMAX-2-0.pcap


Activities
Wireshark
Aug 4 16:05
WIMAX-2-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Num OFDMA Symbols Info
1 0.000000 00:00:00_00:00:00 00:00:00_00:00:00 WiMax 40 0 DL-MAP[Malformed Packet]
2 0.000000 00:00:00_00:00:00 00:00:00_00:00:00 WiMax 42 UL-MAP[Malformed Packet]
3 0.000000 00:00:00_00:00:00 00:00:00_00:00:00 WiMax 79 DCD, DCD TLV error
4 0.000000 00:00:00_00:00:00 00:00:00_00:00:00 WiMax 70 UCD[Malformed Packet]
5 0.010110 00:00:00_00:00:00 00:00:00_00:00:00 WiMax 40 0 DL-MAP[Malformed Packet]
6 0.010110 00:00:00_00:00:00 00:00:00_00:00:00 WiMax 42 UL-MAP[Malformed Packet]
7 0.010110 00:00:00_00:00:00 00:00:00_00:00:00 WiMax 70 UCD[Malformed Packet]
8 0.020221 00:00:00_00:00:00 00:00:00_00:00:00 WiMax 40 0 DL-MAP[Malformed Packet]
9 0.020221 00:00:00_00:00:00 00:00:00_00:00:00 WiMax 42 UL-MAP[Malformed Packet]
Frame 1: 40 bytes on wire (320 bits), 40 bytes captured (320 bits)
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
WiMax Mac to Mac Packet (26 bytes)
PDU (20 bytes) - Generic MAC Header, DL-MAP
[Malformed Packet: WiMax DLMAP]
0000 00 00 00 00 00 00 00 00 00 00 00 08 f0 00 01
0010 00 01 09 14 00 00 14 ff ff 2d 02 01 00 00 00 00
0020 00 03 00 00 0e 00 00 00
WIMAX-2-0.pcap
Packets: 4343 · Displayed: 4343 (100.0%)
Profile: Default


Activities
Wireshark
Aug 4 16:06
Wireshark · Flow · WIMAX-2-0.pcap
00:00:00_00:00:00
0.000000 WiMax: DL-MAP[Malformed Packet]
0.000000 WiMax: UL-MAP[Malformed Packet]
0.000000 WiMax: DCD, DCD TLV error
0.000000 WiMax: UCD[Malformed Packet]
0.010110 WiMax: DL-MAP[Malformed Packet]
0.010110 WiMax: UL-MAP[Malformed Packet]
0.010110 WiMax: UCD[Malformed Packet]
0.020221 WiMax: DL-MAP[Malformed Packet]
0.020221 WiMax: UL-MAP[Malformed Packet]
0.030332 WiMax: DL-MAP[Malformed Packet]
0.030332 WiMax: UL-MAP[Malformed Packet]
0.030332 WiMax: UCD[Malformed Packet]
0.040443 WiMax: DL-MAP[Malformed Packet]
0.040443 WiMax: UL-MAP[Malformed Packet]
0.040443 WiMax: UCD[Malformed Packet]
0.050553 WiMax: DL-MAP[Malformed Packet]
0.050553 WiMax: UL-MAP[Malformed Packet]
0.060664 WiMax: DL-MAP[Malformed Packet]
0.060664 WiMax: UL-MAP[Malformed Packet]
0.060664 WiMax: UCD[Malformed Packet]
0.070775 WiMax: DL-MAP[Malformed Packet]
Packet 18: WiMax: DL-MAP[Malformed Packet]
Limit to display filter
Flow type: All Flows
Addresses: Any
Help
Reset Diagram
Close
Save As...

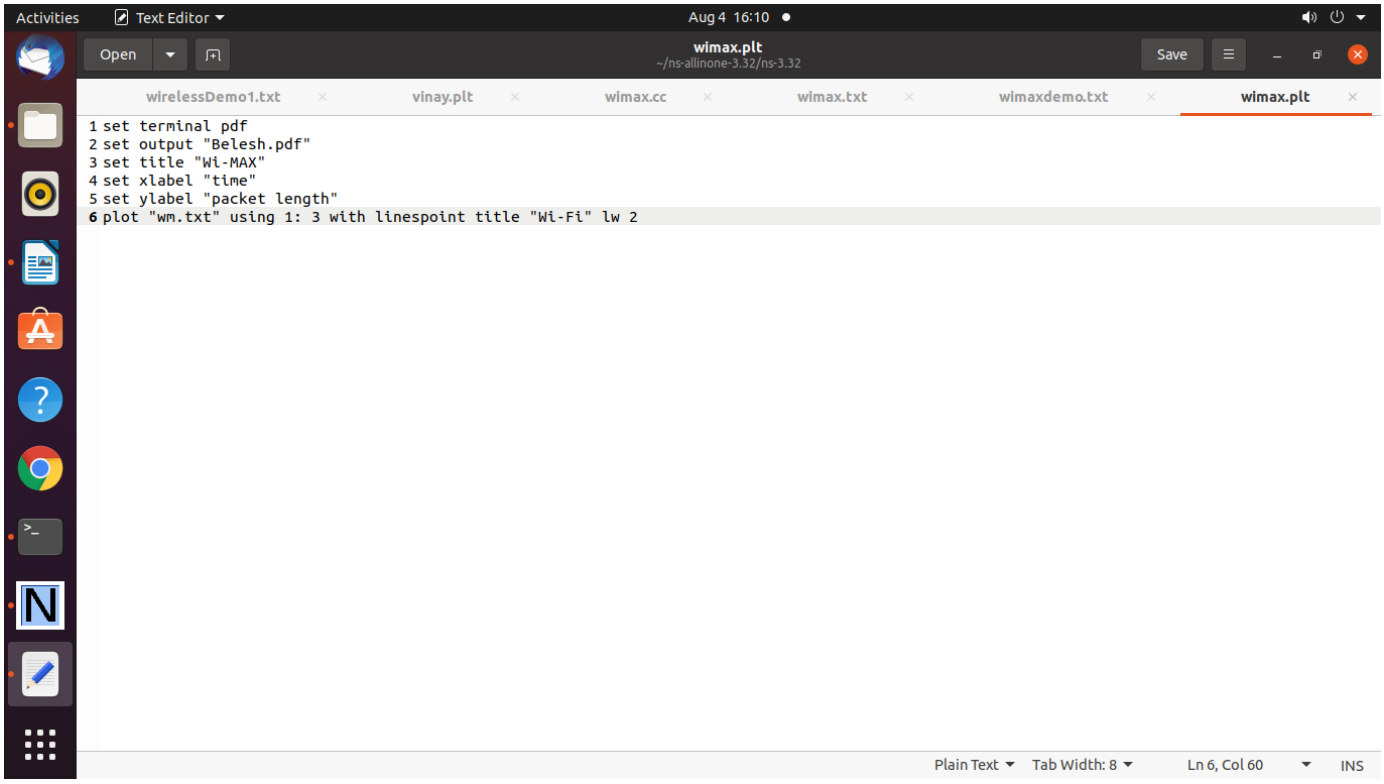
To Plot a Graph Assume following files

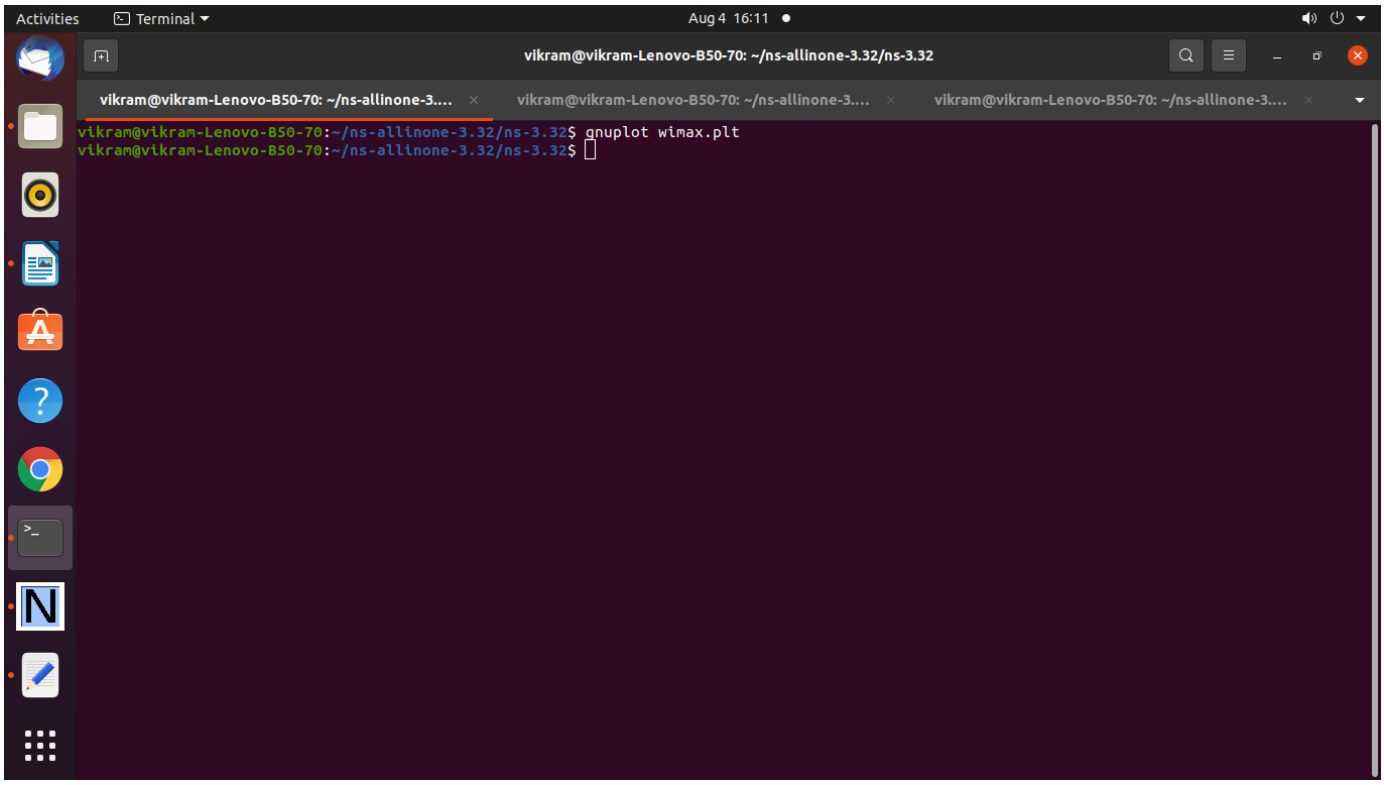
wimaxdemo.txt

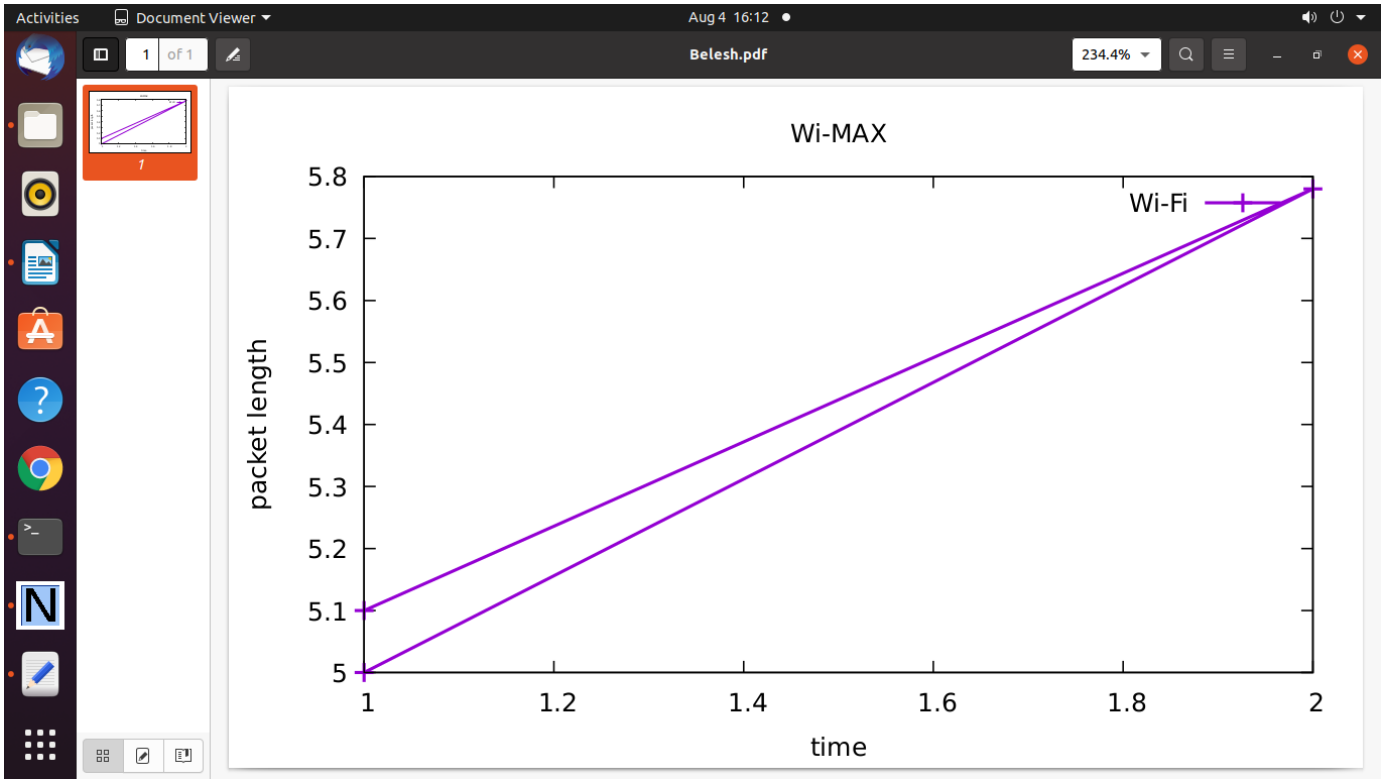


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	40	DL-MAP[Malformed Packet]
2	0.000000	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	42	UL-MAP[Malformed Packet]
3	0.010222	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	40	DL-MAP[Malformed Packet]
4	0.010222	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	42	UL-MAP[Malformed Packet]
5	0.010222	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	70	UCD[Malformed Packet]
6	0.020222	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	40	DL-MAP[Malformed Packet]
7	0.020222	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	42	UL-MAP[Malformed Packet]
8	0.030361	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	40	DL-MAP[Malformed Packet]
9	0.030361	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	51	UL-MAP[Malformed Packet]
10	0.040582	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	40	DL-MAP[Malformed Packet]
11	0.040582	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	42	UL-MAP[Malformed Packet]
12	0.040582	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	79	DCD, DCD TLV error
13	0.050665	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	40	DL-MAP[Malformed Packet]
14	0.050665	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	42	UL-MAP[Malformed Packet]
15	0.050665	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	70	UCD[Malformed Packet]
16	0.060915	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	40	DL-MAP[Malformed Packet]
17	0.060915	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	42	UL-MAP[Malformed Packet]
18	0.060915	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	79	DCD, DCD TLV error
19	0.060915	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	70	UCD[Malformed Packet]
20	0.071054	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	40	DL-MAP[Malformed Packet]
21	0.071054	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	51	UL-MAP[Malformed Packet]
22	0.071054	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	79	DCD, DCD TLV error
23	0.071054	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	70	UCD[Malformed Packet]
24	0.076161	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	36	RNG-REQ, RNG-REQ TLV error
25	0.081025	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	40	DL-MAP[Malformed Packet]
26	0.081025	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	42	UL-MAP[Malformed Packet]
27	0.081025	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	79	DCD, DCD TLV error
28	0.091108	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	40	DL-MAP[Malformed Packet]
29	0.091108	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	42	UL-MAP[Malformed Packet]
30	0.091108	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	70	UCD[Malformed Packet]
31	0.101219	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	40	DL-MAP[Malformed Packet]
32	0.101219	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	42	UL-MAP[Malformed Packet]
33	0.101219	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	70	UCD[Malformed Packet]
34	0.111247	00:00:00_00:00:00	00:00:00_00:00:00	WiMax	40	DL-MAP[Malformed Packet]

And
wimax.plt




Activities
Terminal
Aug 4 16:11
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ gnuplot wimax.plt
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$


Activities
Document Viewer
Aug 4 16:12
1 of 1
Belesh.pdf
234.4%
Wi-MAX
Wi-Fi
packet length
time
5.8
5.7
5.6
5.5
5.4
5.3
5.2
5.1
5
1
1.2
1.4
1.6
1.8
2

==========

Practical 14. Program to simulate IEEE 802.11 Wi-Fi Manager Nodes

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * Copyright (c) 2016 University of Washington
 *
 * 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
 *
 * Authors: Tom Henderson <tomhend@u.washington.edu>
 *          Matías Richart <mrichart@fing.edu.uy>
 *          Sébastien Deronne <sebastien.deronne@gmail.com>
 */

// Test the operation of a wifi manager as the SNR is varied, and create
// a gnuplot output file for plotting.
//
// The test consists of a device acting as server and a device as client generating
traffic.
//
// The output consists of a plot of the rate observed and selected at the client
device.
//
// By default, the 802.11a standard using IdealWifiManager is plotted. Several
command line
// arguments can change the following options:
// --wifiManager (Aarf, Aarfcd, Amrr, Arf, Cara, Ideal, Minstrel, MinstrelHt, Onoe,
Rraa)
// --standard (802.11a, 802.11b, 802.11g, 802.11n-5GHz, 802.11n-2.4GHz,
802.11ac, 802.11-holland, 802.11p-10MHz, 802.11p-5MHz)
// --serverShortGuardInterval and --clientShortGuardInterval (for 802.11n/ac)
// --serverNss and --clientNss (for 802.11n/ac)
// --serverChannelWidth and --clientChannelWidth (for 802.11n/ac)
// --broadcast instead of unicast (default is unicast)
```

```
// --rtsThreshold (by default, value of 99999 disables it)

#include "ns3/log.h"
#include "ns3/config.h"
#include "ns3/uinteger.h"
#include "ns3/boolean.h"
#include "ns3/double.h"
#include "ns3/gnuplot.h"
#include "ns3/command-line.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/ssid.h"
#include "ns3/propagation-loss-model.h"
#include "ns3/propagation-delay-model.h"
#include "ns3/rng-seed-manager.h"
#include "ns3/mobility-helper.h"
#include "ns3/wifi-net-device.h"
#include "ns3/packet-socket-helper.h"
#include "ns3/packet-socket-client.h"
#include "ns3/packet-socket-server.h"
#include "ns3/ht-configuration.h"
#include "ns3/he-configuration.h"

#include "ns3/netanim-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("WifiManagerExample");

// 290K @ 20 MHz
const double NOISE_DBM_Hz = -174.0;
double noiseDbm = NOISE_DBM_Hz;

double g_intervalBytes = 0;
uint64_t g_intervalRate = 0;

void
PacketRx (Ptr<const Packet> pkt, const Address &addr)
{
  g_intervalBytes += pkt->GetSize ();
}

void
RateChange (uint64_t oldVal, uint64_t newVal)
{
  NS_LOG_DEBUG ("Change from " << oldVal << " to " << newVal);
```

```
  g_intervalRate = newVal;
}

/// Step structure
struct Step
{
  double stepSize; ///< step size in dBm
  double stepTime; ///< step size in seconds
};

/// StandardInfo structure
struct StandardInfo
{
  StandardInfo ()
  {
    m_name = "none";
  }
  /**
   * Constructor
   *
   * \param name reference name
   * \param standard wifi standard
   * \param width channel width
   * \param snrLow SNR low
   * \param snrHigh SNR high
   * \param xMin x minimum
   * \param xMax x maximum
   * \param yMax y maximum
   */
  StandardInfo (std::string name, WifiStandard standard, uint16_t width, double
snrLow, double snrHigh, double xMin, double xMax, double yMax)
    : m_name (name),
      m_standard (standard),
      m_width (width),
      m_snrLow (snrLow),
      m_snrHigh (snrHigh),
      m_xMin (xMin),
      m_xMax (xMax),
      m_yMax (yMax)
  {
  }
  std::string m_name; ///< name
  WifiStandard m_standard; ///< standard
  uint16_t m_width; ///< channel width
  double m_snrLow; ///< lowest SNR
```

```
  double m_snrHigh; ///< highest SNR
  double m_xMin;  ///< X minimum
  double m_xMax;  ///< X maximum
  double m_yMax;  ///< Y maximum
};

void
ChangeSignalAndReportRate (Ptr<FixedRssLossModel> rssModel, struct Step
step, double rss, Gnuplot2dDataset& rateDataset, Gnuplot2dDataset&
actualDataset)
{
  NS_LOG_FUNCTION (rssModel << step.stepSize << step.stepTime << rss);
  double snr = rss - noiseDbm;
  rateDataset.Add (snr, g_intervalRate / 1e6);
  // Calculate received rate since last interval
  double currentRate = ((g_intervalBytes * 8) / step.stepTime) / 1e6; // Mb/s
  actualDataset.Add (snr, currentRate);
  rssModel->SetRss (rss - step.stepSize);
  NS_LOG_INFO ("At time " << Simulator::Now ().As (Time::S) << "; selected rate
" << (g_intervalRate / 1e6) << "; observed rate " << currentRate << "; setting new
power to " << rss - step.stepSize);
  g_intervalBytes = 0;
  Simulator::Schedule (Seconds (step.stepTime), &ChangeSignalAndReportRate,
rssModel, step, (rss - step.stepSize), rateDataset, actualDataset);
}

int main (int argc, char *argv[])
{
  std::vector <StandardInfo> serverStandards;
  std::vector <StandardInfo> clientStandards;
  uint32_t steps;
  uint32_t rtsThreshold = 999999;  // disabled even for large A-MPDU
  uint32_t maxAmpduSize = 65535;
  double stepSize = 1; // dBm
  double stepTime = 1; // seconds
  uint32_t packetSize = 1024; // bytes
  bool broadcast = 0;
  int ap1_x = 0;
  int ap1_y = 0;
  int sta1_x = 5;
  int sta1_y = 0;
  uint16_t serverNss = 1;
  uint16_t clientNss = 1;
  uint16_t serverShortGuardInterval = 800;
  uint16_t clientShortGuardInterval = 800;
```

```
  uint16_t serverChannelWidth = 20;
  uint16_t clientChannelWidth = 20;
  std::string wifiManager ("Ideal");
  std::string standard ("802.11a");
  StandardInfo serverSelectedStandard;
  StandardInfo clientSelectedStandard;
  bool infrastructure = false;
  uint32_t maxSlrc = 7;
  uint32_t maxSsrc = 7;

  CommandLine cmd (__FILE__);
  cmd.AddValue ("maxSsrc", "The maximum number of retransmission attempts
for a RTS packet", maxSsrc);
  cmd.AddValue ("maxSlrc", "The maximum number of retransmission attempts for
a Data packet", maxSlrc);
  cmd.AddValue ("rtsThreshold", "RTS threshold", rtsThreshold);
  cmd.AddValue ("maxAmpduSize", "Max A-MPDU size", maxAmpduSize);
  cmd.AddValue ("stepSize", "Power between steps (dBm)", stepSize);
  cmd.AddValue ("stepTime", "Time on each step (seconds)", stepTime);
  cmd.AddValue ("broadcast", "Send broadcast instead of unicast", broadcast);
  cmd.AddValue ("serverChannelWidth", "Set channel width of the server (valid
only for 802.11n or ac)", serverChannelWidth);
  cmd.AddValue ("clientChannelWidth", "Set channel width of the client (valid only
for 802.11n or ac)", clientChannelWidth);
  cmd.AddValue ("serverNss", "Set nss of the server (valid only for 802.11n or ac)",
serverNss);
  cmd.AddValue ("clientNss", "Set nss of the client (valid only for 802.11n or ac)",
clientNss);
  cmd.AddValue ("serverShortGuardInterval", "Set short guard interval of the
server (802.11n/ac/ax) in nanoseconds", serverShortGuardInterval);
  cmd.AddValue ("clientShortGuardInterval", "Set short guard interval of the client
(802.11n/ac/ax) in nanoseconds", clientShortGuardInterval);
  cmd.AddValue ("standard", "Set standard (802.11a, 802.11b, 802.11g, 802.11n-
5GHz, 802.11n-2.4GHz, 802.11ac, 802.11-holland, 802.11p-10MHz, 802.11p-5MHz,
802.11ax-5GHz, 802.11ax-2.4GHz)", standard);
  cmd.AddValue ("wifiManager", "Set wifi rate manager (Aarf, Aarfcd, Amrr, Arf,
Cara, Ideal, Minstrel, MinstrelHt, Onoe, Rraa)", wifiManager);
  cmd.AddValue ("infrastructure", "Use infrastructure instead of adhoc",
infrastructure);
  cmd.Parse (argc,argv);

  // Print out some explanation of what this program does
  std::cout << std::endl << "This program demonstrates and plots the operation of
different " << std::endl;
```

```
  std::cout << "Wi-Fi rate controls on different station configurations," <<
std::endl;
  std::cout << "by stepping down the received signal strength across a wide range"
<< std::endl;
  std::cout << "and observing the adjustment of the rate." << std::endl;
  std::cout << "Run 'wifi-manager-example --PrintHelp' to show program options."
<< std::endl << std::endl;

  if (infrastructure == false)
    {
      NS_ABORT_MSG_IF (serverNss != clientNss, "In ad hoc mode, we assume
sender and receiver are similarly configured");
    }

  if (standard == "802.11b")
    {
      NS_ABORT_MSG_IF (serverChannelWidth != 22 && serverChannelWidth !=
22, "Invalid channel width for standard " << standard);
      NS_ABORT_MSG_IF (serverNss != 1, "Invalid nss for standard " << standard);
      NS_ABORT_MSG_IF (clientChannelWidth != 22 && clientChannelWidth !=
22, "Invalid channel width for standard " << standard);
      NS_ABORT_MSG_IF (clientNss != 1, "Invalid nss for standard " << standard);
    }
  else if (standard == "802.11a" || standard == "802.11g")
    {
      NS_ABORT_MSG_IF (serverChannelWidth != 20, "Invalid channel width for
standard " << standard);
      NS_ABORT_MSG_IF (serverNss != 1, "Invalid nss for standard " << standard);
      NS_ABORT_MSG_IF (clientChannelWidth != 20, "Invalid channel width for
standard " << standard);
      NS_ABORT_MSG_IF (clientNss != 1, "Invalid nss for standard " << standard);
    }
  else if (standard == "802.11n-5GHz" || standard == "802.11n-2.4GHz")
    {
      NS_ABORT_MSG_IF (serverChannelWidth != 20 && serverChannelWidth !=
40, "Invalid channel width for standard " << standard);
      NS_ABORT_MSG_IF (serverNss == 0 || serverNss > 4, "Invalid nss " <<
serverNss << " for standard " << standard);
      NS_ABORT_MSG_IF (clientChannelWidth != 20 && clientChannelWidth !=
40, "Invalid channel width for standard " << standard);
      NS_ABORT_MSG_IF (clientNss == 0 || clientNss > 4, "Invalid nss " <<
clientNss << " for standard " << standard);
    }
  else if (standard == "802.11ac")
    {
```

```
    NS_ABORT_MSG_IF (serverChannelWidth != 20 && serverChannelWidth !=
40 && serverChannelWidth != 80 && serverChannelWidth != 160, "Invalid channel
width for standard " << standard);
    NS_ABORT_MSG_IF (serverNss == 0 || serverNss > 4, "Invalid nss " <<
serverNss << " for standard " << standard);
    NS_ABORT_MSG_IF (clientChannelWidth != 20 && clientChannelWidth !=
40 && clientChannelWidth != 80 && clientChannelWidth != 160, "Invalid channel
width for standard " << standard);
    NS_ABORT_MSG_IF (clientNss == 0 || clientNss > 4, "Invalid nss " <<
clientNss << " for standard " << standard);
   }
  else if (standard == "802.11ax-5GHz" || standard == "802.11ax-2.4GHz")
   {
    NS_ABORT_MSG_IF (serverChannelWidth != 20 && serverChannelWidth !=
40 && serverChannelWidth != 80 && serverChannelWidth != 160, "Invalid channel
width for standard " << standard);
    NS_ABORT_MSG_IF (serverNss == 0 || serverNss > 4, "Invalid nss " <<
serverNss << " for standard " << standard);
    NS_ABORT_MSG_IF (clientChannelWidth != 20 && clientChannelWidth !=
40 && clientChannelWidth != 80 && clientChannelWidth != 160, "Invalid channel
width for standard " << standard);
    NS_ABORT_MSG_IF (clientNss == 0 || clientNss > 4, "Invalid nss " <<
clientNss << " for standard " << standard);
   }

  // As channel width increases, scale up plot's yRange value
  uint32_t channelRateFactor = std::max (clientChannelWidth,
serverChannelWidth) / 20;
  channelRateFactor = channelRateFactor * std::max (clientNss, serverNss);

  // The first number is channel width, second is minimum SNR, third is maximum
  // SNR, fourth and fifth provide xrange axis limits, and sixth the yaxis
  // maximum
  serverStandards.push_back (StandardInfo ("802.11a",
WIFI_STANDARD_80211a, 20, 3, 27, 0, 30, 60));
  serverStandards.push_back (StandardInfo ("802.11b",
WIFI_STANDARD_80211b, 22, -5, 11, -6, 15, 15));
  serverStandards.push_back (StandardInfo ("802.11g",
WIFI_STANDARD_80211g, 20, -5, 27, -6, 30, 60));
  serverStandards.push_back (StandardInfo ("802.11n-5GHz",
WIFI_STANDARD_80211n_5GHZ, serverChannelWidth, 3, 30, 0, 35, 80 *
channelRateFactor));
  serverStandards.push_back (StandardInfo ("802.11n-2.4GHz",
WIFI_STANDARD_80211n_2_4GHZ, serverChannelWidth, 3, 30, 0, 35, 80 *
channelRateFactor));
```

```
  serverStandards.push_back (StandardInfo ("802.11ac",
WIFI_STANDARD_80211ac, serverChannelWidth, 5, 50, 0, 55, 120 *
channelRateFactor));
  serverStandards.push_back (StandardInfo ("802.11-holland",
WIFI_STANDARD_holland, 20, 3, 27, 0, 30, 60));
  serverStandards.push_back (StandardInfo ("802.11p-10MHz",
WIFI_STANDARD_80211p, 10, 3, 27, 0, 30, 60));
  serverStandards.push_back (StandardInfo ("802.11p-5MHz",
WIFI_STANDARD_80211p, 5, 3, 27, 0, 30, 60));
  serverStandards.push_back (StandardInfo ("802.11ax-5GHz",
WIFI_STANDARD_80211ax_5GHZ, serverChannelWidth, 5, 55, 0, 60, 120 *
channelRateFactor));
  serverStandards.push_back (StandardInfo ("802.11ax-2.4GHz",
WIFI_STANDARD_80211ax_2_4GHZ, serverChannelWidth, 5, 55, 0, 60, 120 *
channelRateFactor));

  clientStandards.push_back (StandardInfo ("802.11a",
WIFI_STANDARD_80211a, 20, 3, 27, 0, 30, 60));
  clientStandards.push_back (StandardInfo ("802.11b",
WIFI_STANDARD_80211b, 22, -5, 11, -6, 15, 15));
  clientStandards.push_back (StandardInfo ("802.11g",
WIFI_STANDARD_80211g, 20, -5, 27, -6, 30, 60));
  clientStandards.push_back (StandardInfo ("802.11n-5GHz",
WIFI_STANDARD_80211n_5GHZ, clientChannelWidth, 3, 30, 0, 35, 80 *
channelRateFactor));
  clientStandards.push_back (StandardInfo ("802.11n-2.4GHz",
WIFI_STANDARD_80211n_2_4GHZ, clientChannelWidth, 3, 30, 0, 35, 80 *
channelRateFactor));
  clientStandards.push_back (StandardInfo ("802.11ac",
WIFI_STANDARD_80211ac, clientChannelWidth, 5, 50, 0, 55, 120 *
channelRateFactor));
  clientStandards.push_back (StandardInfo ("802.11-holland",
WIFI_STANDARD_holland, 20, 3, 27, 0, 30, 60));
  clientStandards.push_back (StandardInfo ("802.11p-10MHz",
WIFI_STANDARD_80211p, 10, 3, 27, 0, 30, 60));
  clientStandards.push_back (StandardInfo ("802.11p-5MHz",
WIFI_STANDARD_80211p, 5, 3, 27, 0, 30, 60));
  clientStandards.push_back (StandardInfo ("802.11ax-5GHz",
WIFI_STANDARD_80211ax_5GHZ, clientChannelWidth, 5, 55, 0, 60, 160 *
channelRateFactor));
  clientStandards.push_back (StandardInfo ("802.11ax-2.4GHz",
WIFI_STANDARD_80211ax_2_4GHZ, clientChannelWidth, 5, 55, 0, 60, 160 *
channelRateFactor));

  for (std::vector<StandardInfo>::size_type i = 0; i != serverStandards.size (); i++)
```

```
  {
    if (standard == serverStandards[i].m_name)
      {
        serverSelectedStandard = serverStandards[i];
      }
  }
 for (std::vector<StandardInfo>::size_type i = 0; i != clientStandards.size (); i++)
  {
    if (standard == clientStandards[i].m_name)
      {
        clientSelectedStandard = clientStandards[i];
      }
  }

 NS_ABORT_MSG_IF (serverSelectedStandard.m_name == "none", "Standard "
<< standard << " not found");
 NS_ABORT_MSG_IF (clientSelectedStandard.m_name == "none", "Standard "
<< standard << " not found");
 std::cout << "Testing " << serverSelectedStandard.m_name << " with " <<
wifiManager << " ..." << std::endl;
 NS_ABORT_MSG_IF (clientSelectedStandard.m_snrLow >=
clientSelectedStandard.m_snrHigh, "SNR values in wrong order");
 steps = static_cast<uint32_t> (std::abs (static_cast<double>
(clientSelectedStandard.m_snrHigh - clientSelectedStandard.m_snrLow ) /
stepSize) + 1);
 NS_LOG_DEBUG ("Using " << steps << " steps for SNR range " <<
clientSelectedStandard.m_snrLow << ":" << clientSelectedStandard.m_snrHigh);
 Ptr<Node> clientNode = CreateObject<Node> ();
 Ptr<Node> serverNode = CreateObject<Node> ();

 std::string plotName = "wifi-manager-example-";
 std::string dataName = "wifi-manager-example-";
 plotName += wifiManager;
 dataName += wifiManager;
 plotName += "-";
 dataName += "-";
 plotName += standard;
 dataName += standard;
 if (standard == "802.11n-5GHz"
    || standard == "802.11n-2.4GHz"
    || standard == "802.11ac"
    || standard == "802.11ax-5GHz"
    || standard == "802.11ax-2.4GHz")
  {
    plotName += "-server_";
```

```
    dataName += "-server_";
    std::ostringstream oss;
    oss << serverChannelWidth << "MHz_" << serverShortGuardInterval << "ns_"
<< serverNss << "SS";
    plotName += oss.str ();
    dataName += oss.str ();
    plotName += "-client_";
    dataName += "-client_";
    oss.str ("");
    oss << clientChannelWidth << "MHz_" << clientShortGuardInterval << "ns_"
<< clientNss << "SS";
    plotName += oss.str ();
    dataName += oss.str ();
   }
  plotName += ".eps";
  dataName += ".plt";
  std::ofstream outfile (dataName.c_str ());
  Gnuplot gnuplot = Gnuplot (plotName);

  Config::SetDefault ("ns3::WifiRemoteStationManager::MaxSlrc", UintegerValue
(maxSlrc));
  Config::SetDefault ("ns3::WifiRemoteStationManager::MaxSsrc", UintegerValue
(maxSsrc));
  Config::SetDefault ("ns3::MinstrelWifiManager::PrintStats", BooleanValue
(true));
  Config::SetDefault ("ns3::MinstrelWifiManager::PrintSamples", BooleanValue
(true));
  Config::SetDefault ("ns3::MinstrelHtWifiManager::PrintStats", BooleanValue
(true));

  WifiHelper wifi;
  wifi.SetStandard (serverSelectedStandard.m_standard);
  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();

  Ptr<YansWifiChannel> wifiChannel = CreateObject<YansWifiChannel> ();
  Ptr<ConstantSpeedPropagationDelayModel> delayModel =
CreateObject<ConstantSpeedPropagationDelayModel> ();
  wifiChannel->SetPropagationDelayModel (delayModel);
  Ptr<FixedRssLossModel> rssLossModel = CreateObject<FixedRssLossModel> ();
  wifiChannel->SetPropagationLossModel (rssLossModel);
  wifiPhy.SetChannel (wifiChannel);

  wifi.SetRemoteStationManager ("ns3::" + wifiManager + "WifiManager",
"RtsCtsThreshold", UintegerValue (rtsThreshold));
```

```
  NetDeviceContainer serverDevice;
  NetDeviceContainer clientDevice;

  WifiMacHelper wifiMac;
  if (infrastructure)
    {
      Ssid ssid = Ssid ("ns-3-ssid");
      wifiMac.SetType ("ns3::StaWifiMac",
                       "Ssid", SsidValue (ssid));
      serverDevice = wifi.Install (wifiPhy, wifiMac, serverNode);
      wifiMac.SetType ("ns3::ApWifiMac",
                       "Ssid", SsidValue (ssid));
      clientDevice = wifi.Install (wifiPhy, wifiMac, clientNode);
    }
  else
    {
      wifiMac.SetType ("ns3::AdhocWifiMac");
      serverDevice = wifi.Install (wifiPhy, wifiMac, serverNode);
      clientDevice = wifi.Install (wifiPhy, wifiMac, clientNode);
    }

  RngSeedManager::SetSeed (1);
  RngSeedManager::SetRun (2);
  wifi.AssignStreams (serverDevice, 100);
  wifi.AssignStreams (clientDevice, 100);

  Config::Set
("/NodeList/*/DeviceList/*/$ns3::WifiNetDevice/Mac/BE_MaxAmpduSize",
UintegerValue (maxAmpduSize));

  Config::ConnectWithoutContext
("/NodeList/0/DeviceList/*/$ns3::WifiNetDevice/RemoteStationManager/$ns3::"
+ wifiManager + "WifiManager/Rate", MakeCallback (&RateChange));

  // Configure the mobility.
  MobilityHelper mobility;
  Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator>
();
  //Initial position of AP and STA
  positionAlloc->Add (Vector (ap1_x, ap1_y, 0.0));
  NS_LOG_INFO ("Setting initial AP position to " << Vector (ap1_x, ap1_y, 0.0));
  positionAlloc->Add (Vector (sta1_x, sta1_y, 0.0));
  NS_LOG_INFO ("Setting initial STA position to " << Vector (sta1_x, sta1_y,
0.0));
  mobility.SetPositionAllocator (positionAlloc);
```

```
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility.Install (clientNode);
  mobility.Install (serverNode);

  AnimationInterface anim("wi-fi.xml");

  AnimationInterface::SetConstantPosition (serverNode, 10, 30);
  AnimationInterface::SetConstantPosition (clientNode, 10, 45);




  Gnuplot2dDataset rateDataset (clientSelectedStandard.m_name + std::string ("-
rate selected"));
  Gnuplot2dDataset actualDataset (clientSelectedStandard.m_name + std::string
("-observed"));
  struct Step step;
  step.stepSize = stepSize;
  step.stepTime = stepTime;

  // Perform post-install configuration from defaults for channel width,
  // guard interval, and nss, if necessary
  // Obtain pointer to the WifiPhy
  Ptr<NetDevice> ndClient = clientDevice.Get (0);
  Ptr<NetDevice> ndServer = serverDevice.Get (0);
  Ptr<WifiNetDevice> wndClient = ndClient->GetObject<WifiNetDevice> ();
  Ptr<WifiNetDevice> wndServer = ndServer->GetObject<WifiNetDevice> ();
  Ptr<WifiPhy> wifiPhyPtrClient = wndClient->GetPhy ();
  Ptr<WifiPhy> wifiPhyPtrServer = wndServer->GetPhy ();
  uint8_t t_clientNss = static_cast<uint8_t> (clientNss);
  uint8_t t_serverNss = static_cast<uint8_t> (serverNss);
  wifiPhyPtrClient->SetNumberOfAntennas (t_clientNss);
  wifiPhyPtrClient->SetMaxSupportedTxSpatialStreams (t_clientNss);
  wifiPhyPtrClient->SetMaxSupportedRxSpatialStreams (t_clientNss);
  wifiPhyPtrServer->SetNumberOfAntennas (t_serverNss);
  wifiPhyPtrServer->SetMaxSupportedTxSpatialStreams (t_serverNss);
  wifiPhyPtrServer->SetMaxSupportedRxSpatialStreams (t_serverNss);
  // Only set the channel width and guard interval for HT and VHT modes
  if (serverSelectedStandard.m_name == "802.11n-5GHz"
      || serverSelectedStandard.m_name == "802.11n-2.4GHz"
      || serverSelectedStandard.m_name == "802.11ac")
    {
      wifiPhyPtrServer->SetChannelWidth (serverSelectedStandard.m_width);
      wifiPhyPtrClient->SetChannelWidth (clientSelectedStandard.m_width);
```

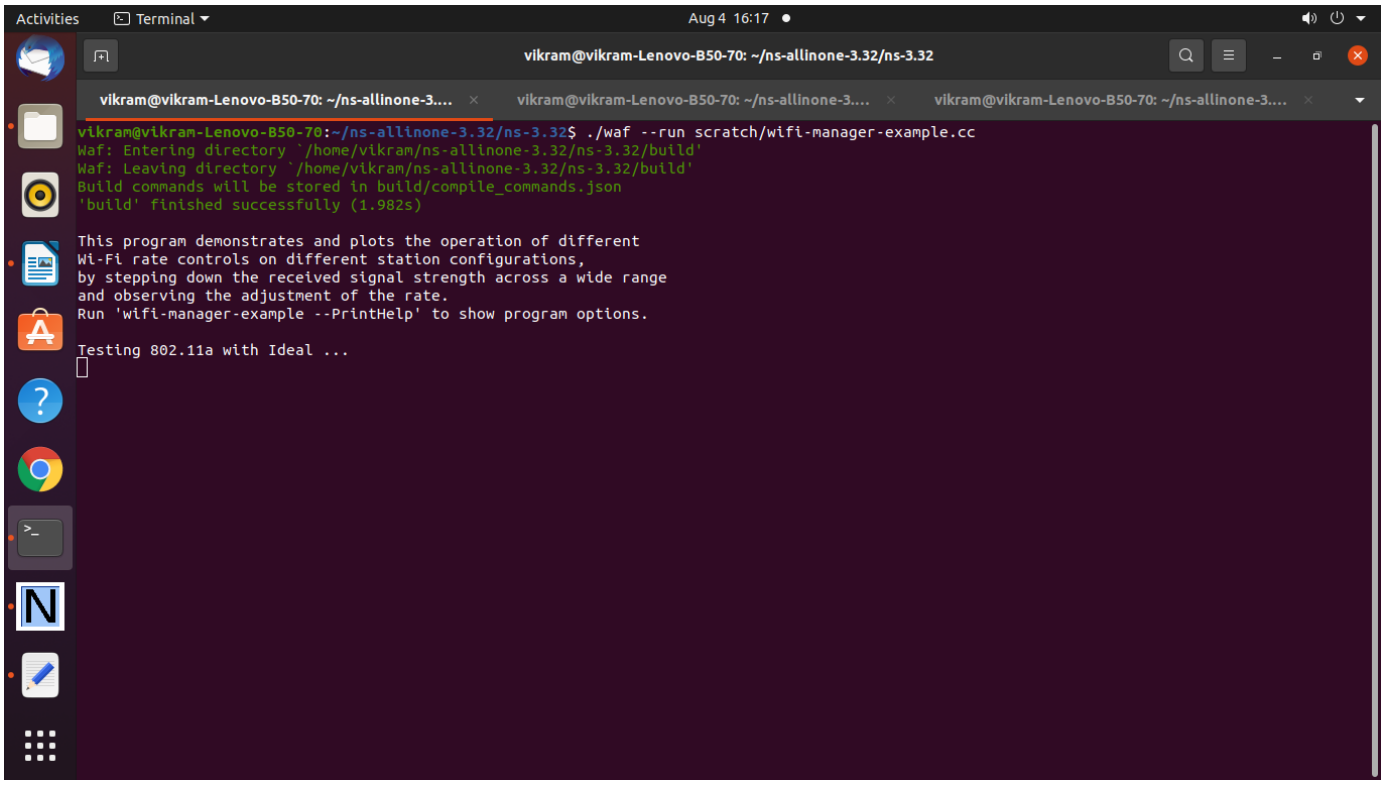
```
    Ptr<HtConfiguration> clientHtConfiguration =
wndClient->GetHtConfiguration ();
    clientHtConfiguration->SetShortGuardIntervalSupported
(clientShortGuardInterval == 400);
    Ptr<HtConfiguration> serverHtConfiguration =
wndServer->GetHtConfiguration ();
    serverHtConfiguration->SetShortGuardIntervalSupported
(serverShortGuardInterval == 400);
   }
  else if (serverSelectedStandard.m_name == "802.11ax-5GHz"
        || serverSelectedStandard.m_name == "802.11ax-2.4GHz")
   {
    wifiPhyPtrServer->SetChannelWidth (serverSelectedStandard.m_width);
    wifiPhyPtrClient->SetChannelWidth (clientSelectedStandard.m_width);
    wndServer->GetHeConfiguration ()->SetGuardInterval (NanoSeconds
(clientShortGuardInterval));
    wndClient->GetHeConfiguration ()->SetGuardInterval (NanoSeconds
(clientShortGuardInterval));
   }
  NS_LOG_DEBUG ("Channel width " << wifiPhyPtrClient->GetChannelWidth ()
<< " noiseDbm " << noiseDbm);
  NS_LOG_DEBUG ("NSS " <<
wifiPhyPtrClient->GetMaxSupportedTxSpatialStreams ());

  // Configure signal and noise, and schedule first iteration
  noiseDbm += 10 * log10 (clientSelectedStandard.m_width * 1000000);
  double rssCurrent = (clientSelectedStandard.m_snrHigh + noiseDbm);
  rssLossModel->SetRss (rssCurrent);
  NS_LOG_INFO ("Setting initial Rss to " << rssCurrent);
  //Move the STA by stepsSize meters every stepTime seconds
  Simulator::Schedule (Seconds (0.5 + stepTime), &ChangeSignalAndReportRate,
rssLossModel, step, rssCurrent, rateDataset, actualDataset);

  PacketSocketHelper packetSocketHelper;
  packetSocketHelper.Install (serverNode);
  packetSocketHelper.Install (clientNode);

  PacketSocketAddress socketAddr;
  socketAddr.SetSingleDevice (serverDevice.Get (0)->GetIfIndex ());
  if (broadcast)
   {
    socketAddr.SetPhysicalAddress (serverDevice.Get (0)->GetBroadcast ());
   }
  else
   {
```

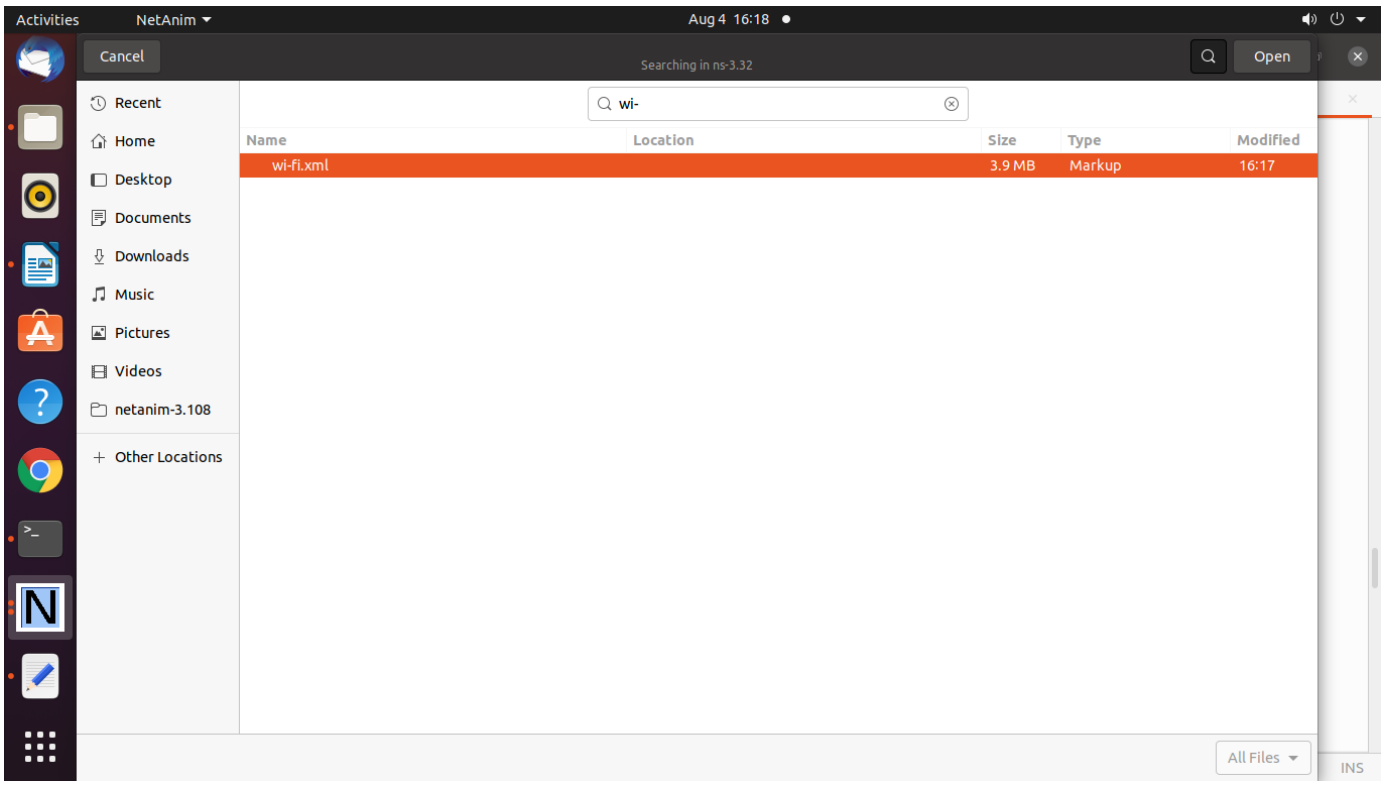
```
    socketAddr.SetPhysicalAddress (serverDevice.Get (0)->GetAddress ());
  }
// Arbitrary protocol type.
// Note: PacketSocket doesn't have any L4 multiplexing or demultiplexing
//       The only mux/demux is based on the protocol field
socketAddr.SetProtocol (1);

Ptr<PacketSocketClient> client = CreateObject<PacketSocketClient> ();
client->SetRemote (socketAddr);
client->SetStartTime (Seconds (0.5));  // allow simulation warmup
client->SetAttribute ("MaxPackets", UintegerValue (0));  // unlimited
client->SetAttribute ("PacketSize", UintegerValue (packetSize));

// Set a maximum rate 10% above the yMax specified for the selected standard
double rate = clientSelectedStandard.m_yMax * 1e6 * 1.10;
double clientInterval = static_cast<double> (packetSize) * 8 / rate;
NS_LOG_DEBUG ("Setting interval to " << clientInterval << " sec for rate of " <<
rate << " bits/sec");

client->SetAttribute ("Interval", TimeValue (Seconds (clientInterval)));
clientNode->AddApplication (client);

Ptr<PacketSocketServer> server = CreateObject<PacketSocketServer> ();
server->SetLocal (socketAddr);
server->TraceConnectWithoutContext ("Rx", MakeCallback (&PacketRx));
serverNode->AddApplication (server);

Simulator::Stop (Seconds ((steps + 1) * stepTime));
Simulator::Run ();
Simulator::Destroy ();

gnuplot.AddDataset (rateDataset);
gnuplot.AddDataset (actualDataset);

std::ostringstream xMinStr, xMaxStr, yMaxStr;
std::string xRangeStr ("set xrange [");
xMinStr << clientSelectedStandard.m_xMin;
xRangeStr.append (xMinStr.str ());
xRangeStr.append (":");
xMaxStr << clientSelectedStandard.m_xMax;
xRangeStr.append (xMaxStr.str ());
xRangeStr.append ("]");
std::string yRangeStr ("set yrange [0:");
yMaxStr << clientSelectedStandard.m_yMax;
yRangeStr.append (yMaxStr.str ());
```

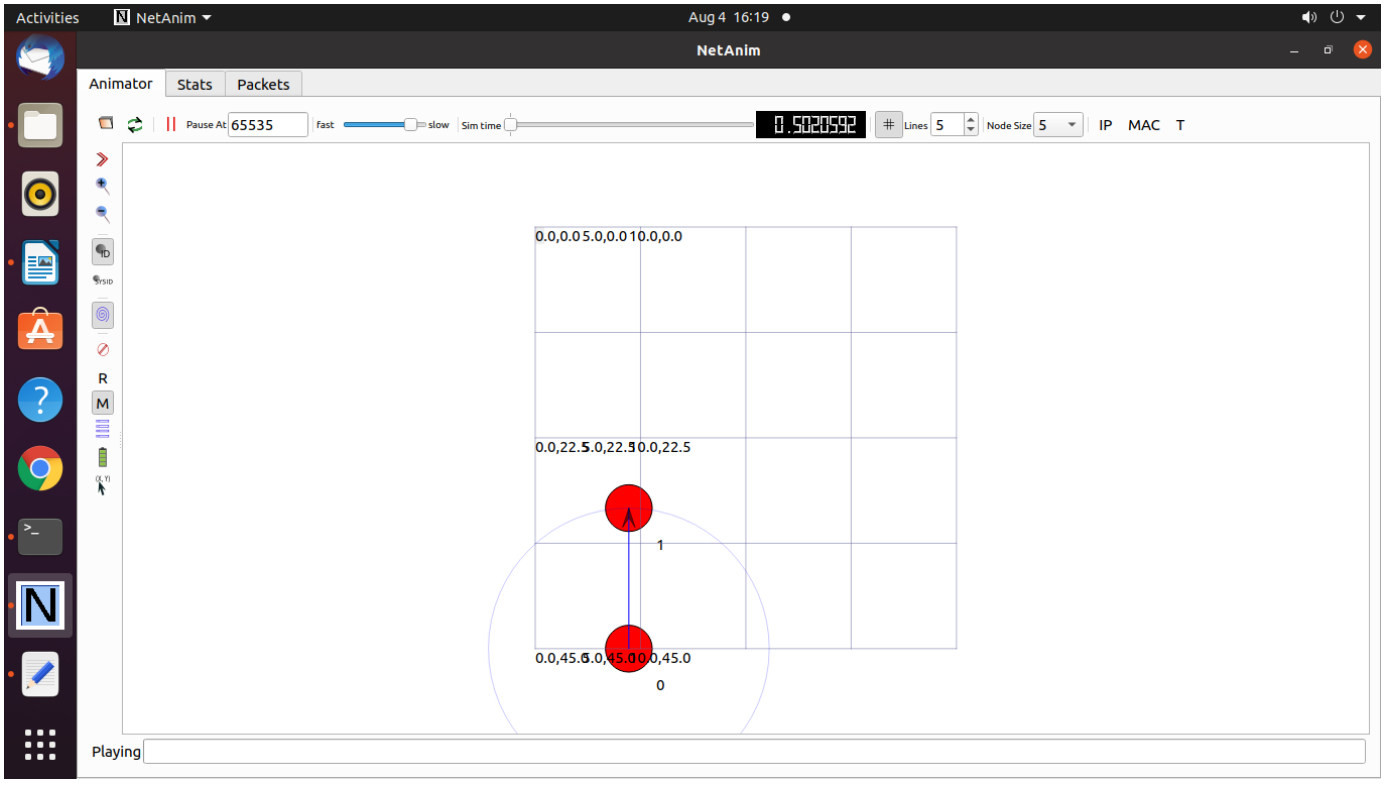
```
yRangeStr.append ("]");

std::string title ("Results for ");
title.append (standard);
title.append (" with ");
title.append (wifiManager);
title.append ("\\n");
if (standard == "802.11n-5GHz"
    || standard == "802.11n-2.4GHz"
    || standard == "802.11ac"
    || standard == "802.11n-5GHz"
    || standard == "802.11ax-2.4GHz")
  {
    std::ostringstream serverGiStrStr;
    std::ostringstream serverWidthStrStr;
    std::ostringstream serverNssStrStr;
    title.append ("server: width=");
    serverWidthStrStr << serverSelectedStandard.m_width;
    title.append (serverWidthStrStr.str ());
    title.append ("MHz");
    title.append (" GI=");
    serverGiStrStr << serverShortGuardInterval;
    title.append (serverGiStrStr.str ());
    title.append ("ns");
    title.append (" nss=");
    serverNssStrStr << serverNss;
    title.append (serverNssStrStr.str ());
    title.append ("\\n");
    std::ostringstream clientGiStrStr;
    std::ostringstream clientWidthStrStr;
    std::ostringstream clientNssStrStr;
    title.append ("client: width=");
    clientWidthStrStr << clientSelectedStandard.m_width;
    title.append (clientWidthStrStr.str ());
    title.append ("MHz");
    title.append (" GI=");
    clientGiStrStr << clientShortGuardInterval;
    title.append (clientGiStrStr.str ());
    title.append ("ns");
    title.append (" nss=");
    clientNssStrStr << clientNss;
    title.append (clientNssStrStr.str ());
  }
gnuplot.SetTerminal ("postscript eps color enh \"Times-BoldItalic\"");
gnuplot.SetLegend ("SNR (dB)", "Rate (Mb/s)");
```

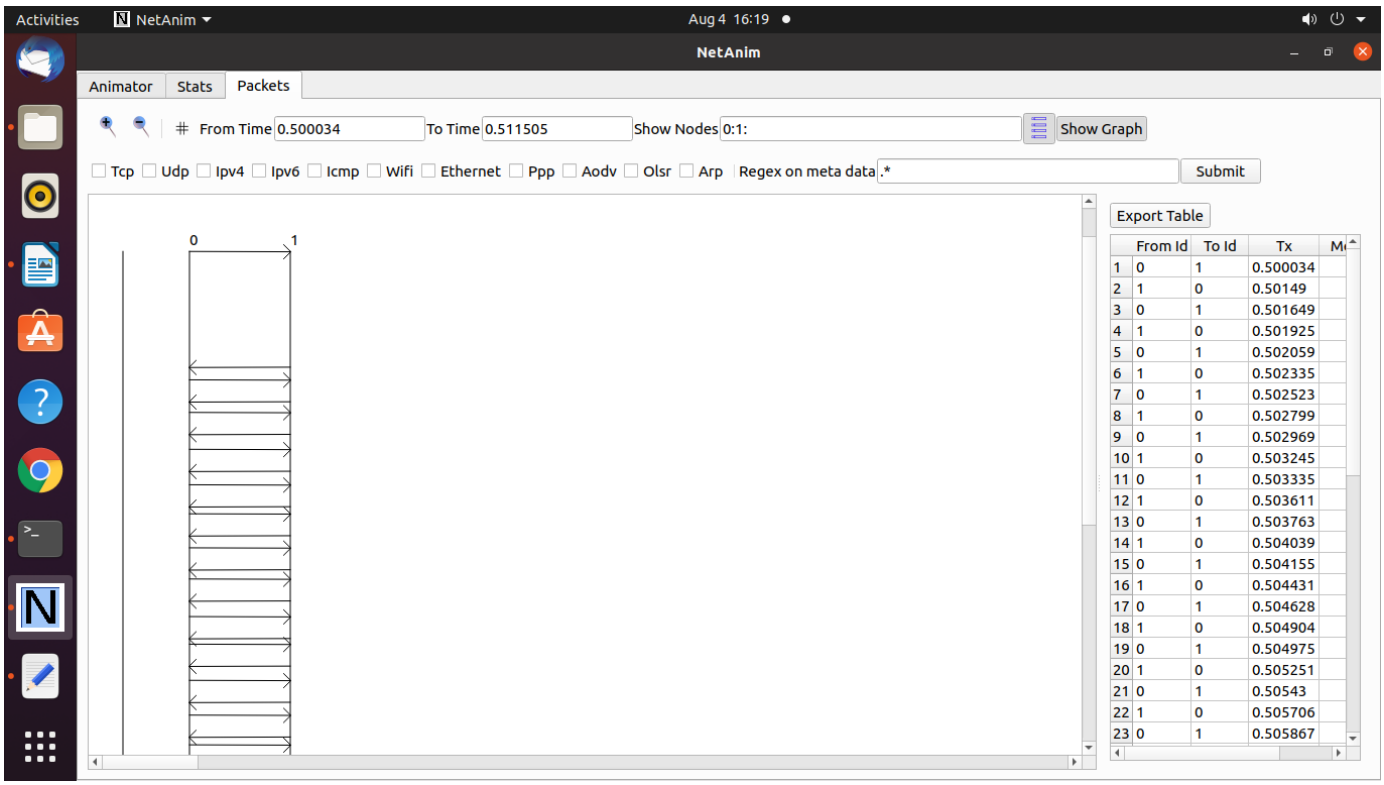
```
  gnuplot.SetTitle (title);
  gnuplot.SetExtra (xRangeStr);
  gnuplot.AppendExtra (yRangeStr);
  gnuplot.AppendExtra ("set key top left");
  gnuplot.GenerateOutput (outfile);
  outfile.close ();

  return 0;
}
```

Output :




Activities
NetAnim
Aug 4 16:18
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
wi-
Name
Location
Size
Type
Modified
wi-fi.xml
3.9 MB
Markup
16:17
All Files
INS


Activities
NetAnim
Aug 4 16:19
NetAnim
Animator
Stats
Packets
Pause At
65535
fast
slow
Sim time
Lines
5
Node Size
5
IP
MAC
T
0.0,0.0
5.0,0.0
10.0,0.0
0.0,22.5
5.0,22.5
10.0,22.5
1
0.0,45.0
5.0,45.0
10.0,45.0
0
Playing

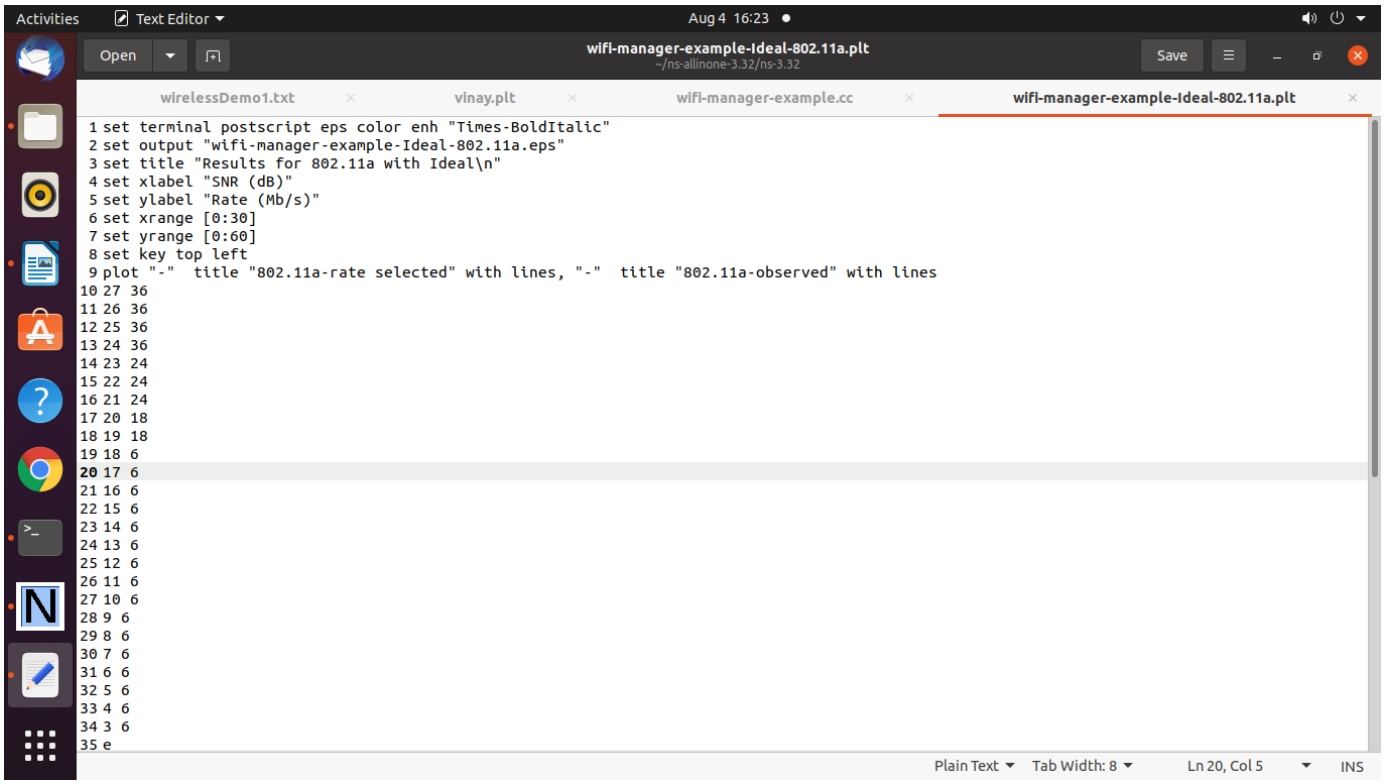


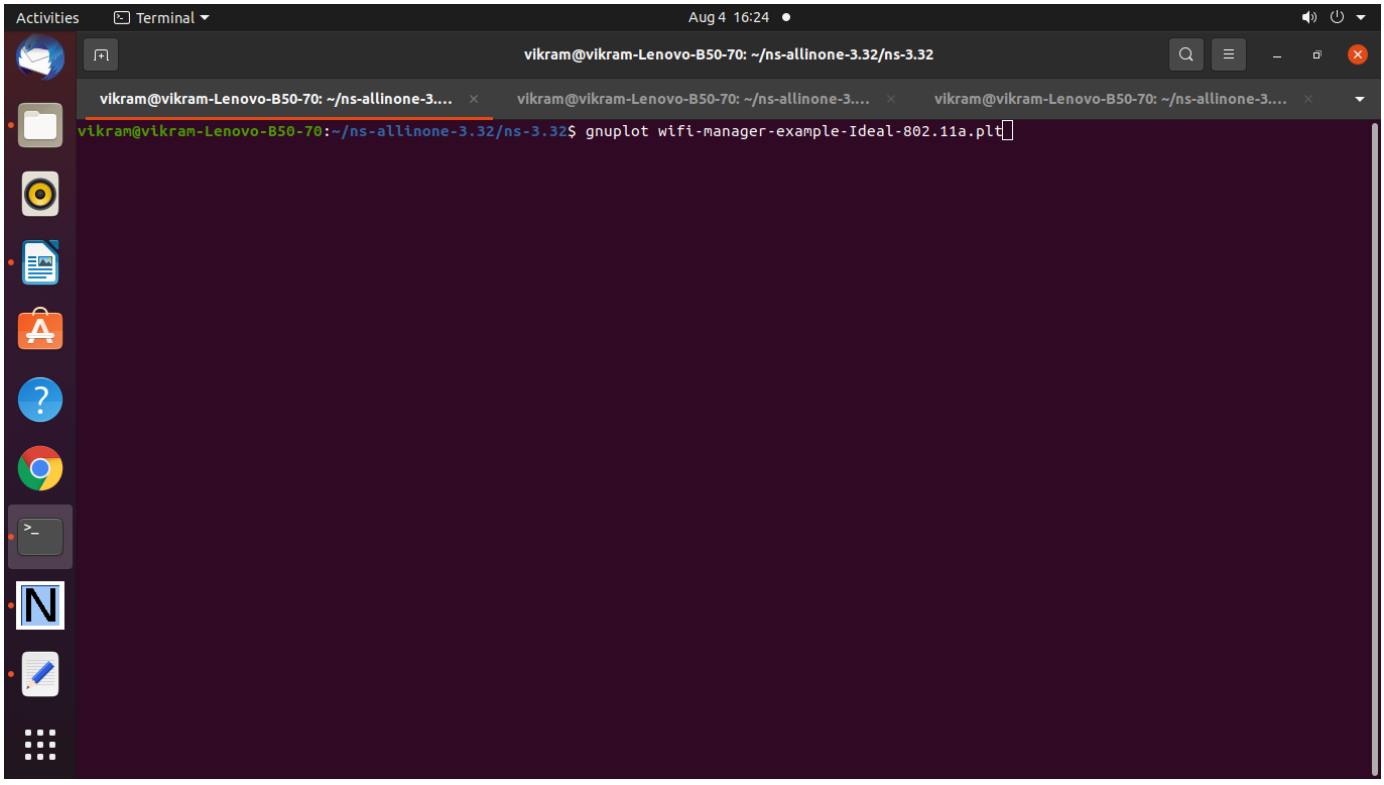
To Draw the Graph consider the following file

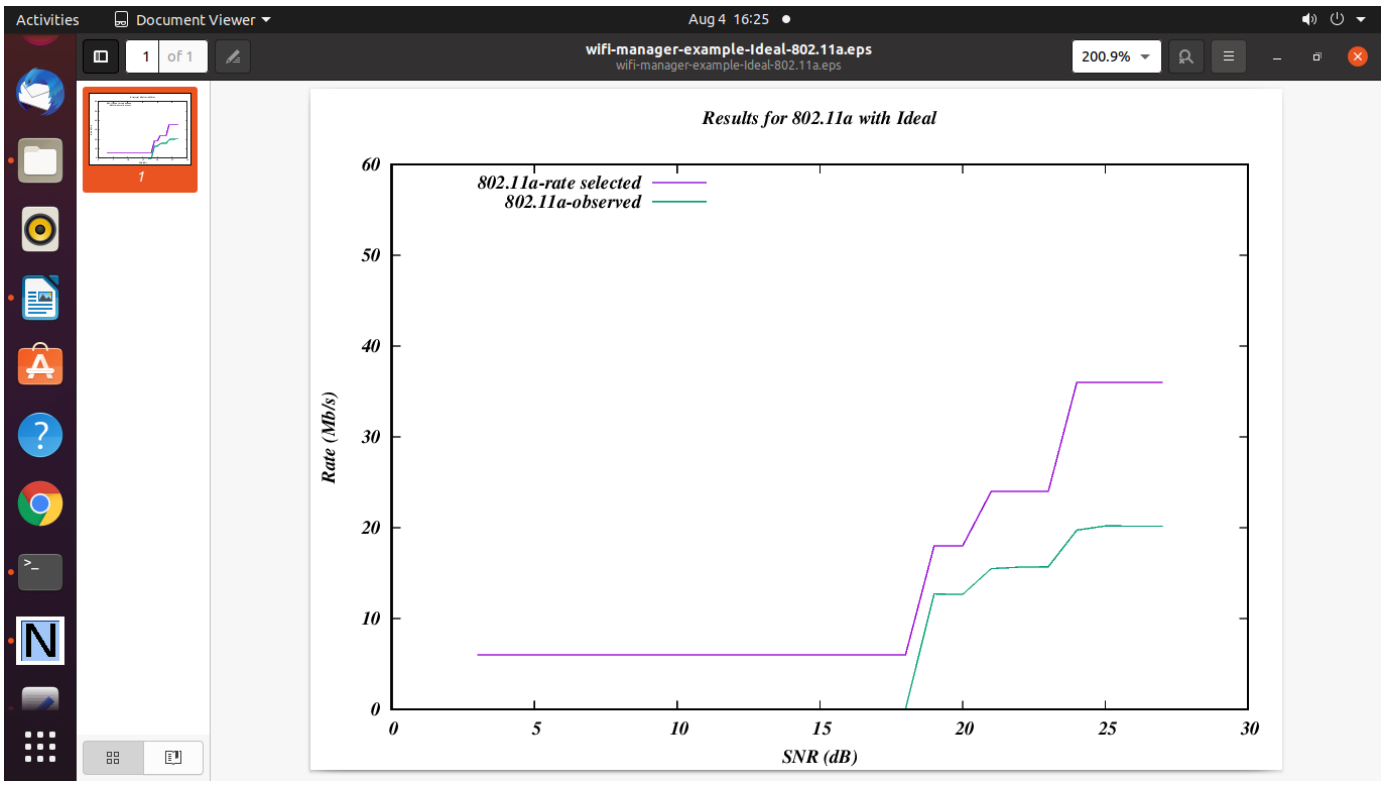
wifi-manager-example-Ideal-802.11a.plt

```
set terminal postscript eps color enh "Times-BoldItalic"
set output "wifi-manager-example-Ideal-802.11a.eps"
set title "Results for 802.11a with Ideal\n"
set xlabel "SNR (dB)"
set ylabel "Rate (Mb/s)"
set xrange [0:30]
set yrange [0:60]
set key top left
plot "-"  title "802.11a-rate selected" with lines, "-"  title "802.11a-observed" with lines
27 36
26 36
25 36
24 36
23 24
22 24
21 24
20 18
19 18
18 6
```

17 6
16 6
15 6
14 6
13 6
12 6
11 6
10 6
9 6
8 6
7 6
6 6
5 6
4 6
3 6
e
27 20.1851
26 20.1851
25 20.2015
24 19.7345
23 15.7041
22 15.6631
21 15.4993
20 12.6566
19 12.714
18 0.008192
17 0
16 0
15 0
14 0
13 0
12 0
11 0
10 0
9 0
8 0
7 0
6 0
5 0
4 0
3 0
e


Activities
Text Editor
Aug 4 16:23
Open
wifi-manager-example-Ideal-802.11a.plt
~/ns-allinone-3.32/ns-3.32
Save
wirelessDemo1.txt
vinay.plt
wifi-manager-example.cc
wifi-manager-example-Ideal-802.11a.plt
1 set terminal postscript eps color enh "Times-BoldItalic"
2 set output "wifi-manager-example-Ideal-802.11a.eps"
3 set title "Results for 802.11a with Ideal\n"
4 set xlabel "SNR (dB)"
5 set ylabel "Rate (Mb/s)"
6 set xrange [0:30]
7 set yrange [0:60]
8 set key top left
9 plot "-" title "802.11a-rate selected" with lines, "-" title "802.11a-observed" with lines
10 27 36
11 26 36
12 25 36
13 24 36
14 23 24
15 22 24
16 21 24
17 20 18
18 19 18
19 18 6
20 17 6
21 16 6
22 15 6
23 14 6
24 13 6
25 12 6
26 11 6
27 10 6
28 9 6
29 8 6
30 7 6
31 6 6
32 5 6
33 4 6
34 3 6
35 e
Plain Text
Tab Width: 8
Ln 20, Col 5
INS


Activities
Terminal
Aug 4 16:24
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ gnuplot wifi-manager-example-Ideal-802.11a.plt


Activities
Document Viewer
Aug 4 16:25
1 of 1
wifi-manager-example-Ideal-802.11a.eps
wifi-manager-example-Ideal-802.11a.eps
200.9%
Results for 802.11a with Ideal
802.11a-rate selected
802.11a-observed
Rate (Mb/s)
SNR (dB)
0
10
20
30
40
50
60
0
5
10
15
20
25
30

==========

Practical 16 : Program to Simulatr Wi-Fi nodes

test.cc

```
#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include <string.h>
#include <fstream>
#include <iterator>

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/internet-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/energy-module.h"
#include "ns3/wifi-radio-energy-model-helper.h"
#include "ns3/config-store-module.h"
#include "ns3/netanim-module.h"
#include "ns3/energy-source-container.h"

using namespace ns3;
using namespace std;

NS_LOG_COMPONENT_DEFINE ("Test1");

static void
GenerateTraffic (Ptr<Socket> socket, uint32_t pktSize, uint32_t pktCount,
                 Time pktInterval)
{
  if(pktCount > 0)
  {
    socket->Send (Create<Packet> (pktSize));
    Simulator::Schedule(pktInterval, &GenerateTraffic, socket, pktSize,
                        pktCount - 1, pktInterval);
  }
}

void EnergyPrint(EnergySourceContainer source, int rounds)
{
    int id = 0;
    int k = 1;
    double remain = 50;
```

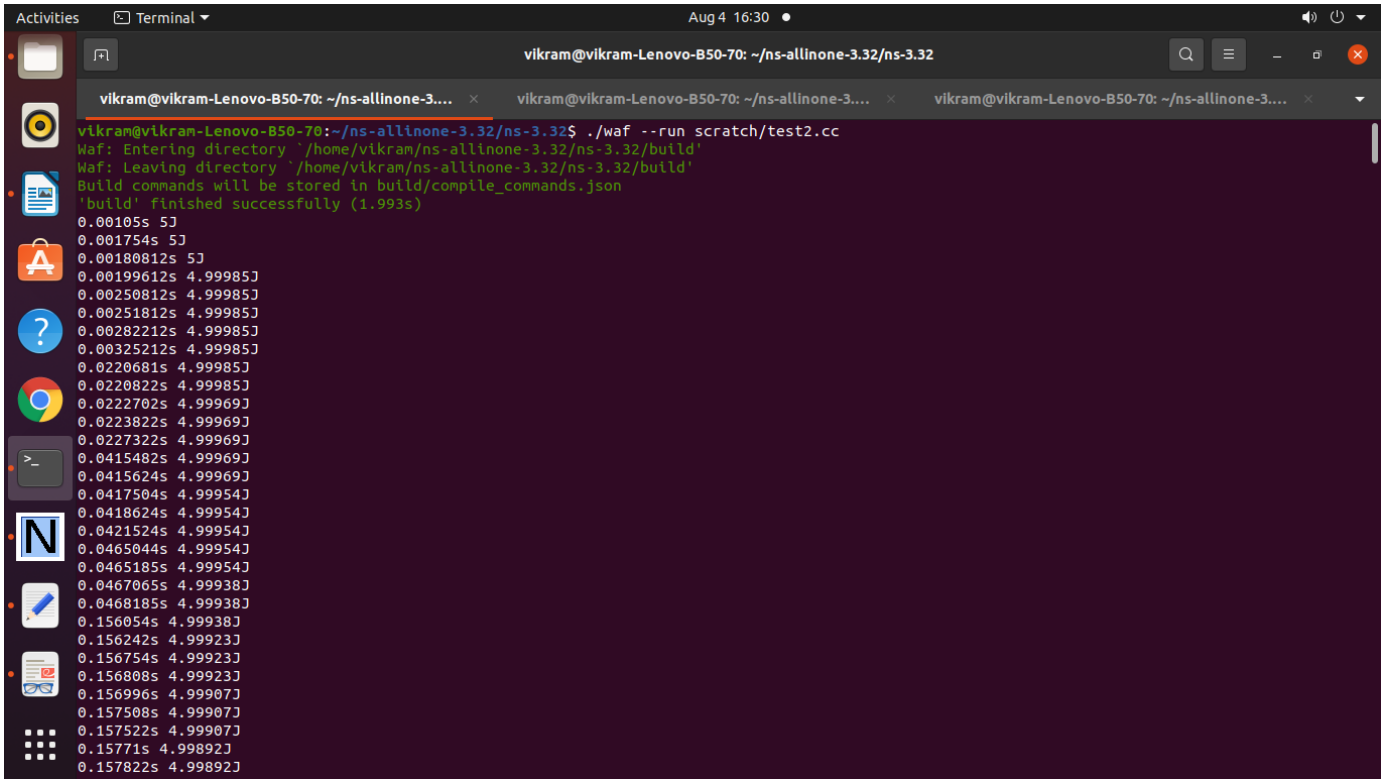
```
   for (EnergySourceContainer::Iterator c = source.Begin(); c != source.End(); c++)
   {
    double energyRemain = (*c)->GetRemainingEnergy();
    if (energyRemain < remain)
    {
     remain = energyRemain;
     id = k;
    }
     NS_LOG_UNCOND("Node "<<k<<":"<<energyRemain<<"J");
    k++;
   }
   NS_LOG_UNCOND("Rounds "<<rounds<<": Node "<<id<<" lowest energy
"<<remain<<"J");
}

void SendPacket(Ptr<Socket> source[], int Cluster_Size, double start,
Ptr<DeviceEnergyModel> modelPtr)
{
   double interval = 0.1;
   uint32_t PpacketSize = 5000;
   uint32_t numPackets = 1;
   Time interPacketInterval = Seconds (interval);

   for (int i = 0; i < Cluster_Size; i++)
   {
    if (i == 1)
    {
     modelPtr->SetAttribute("RxCurrentA", DoubleValue (0.45));
     modelPtr->SetAttribute("TxCurrentA", DoubleValue (0.21));
     modelPtr->SetAttribute("IdleCurrentA", DoubleValue (0.1));
     modelPtr->SetAttribute("SleepCurrentA", DoubleValue (0.03));
    }
    Time time = Seconds(start + i*interval*1.5);
    Simulator::Schedule (time, &GenerateTraffic, source[i], PpacketSize,
    numPackets, interPacketInterval);
    if (i == 1)
    {
     modelPtr->SetAttribute("RxCurrentA", DoubleValue (0.00001));
     modelPtr->SetAttribute("TxCurrentA", DoubleValue (0.00001));
     modelPtr->SetAttribute("IdleCurrentA", DoubleValue (0.00001));
     modelPtr->SetAttribute("SleepCurrentA", DoubleValue (0.00001));
    }
   }
}
```

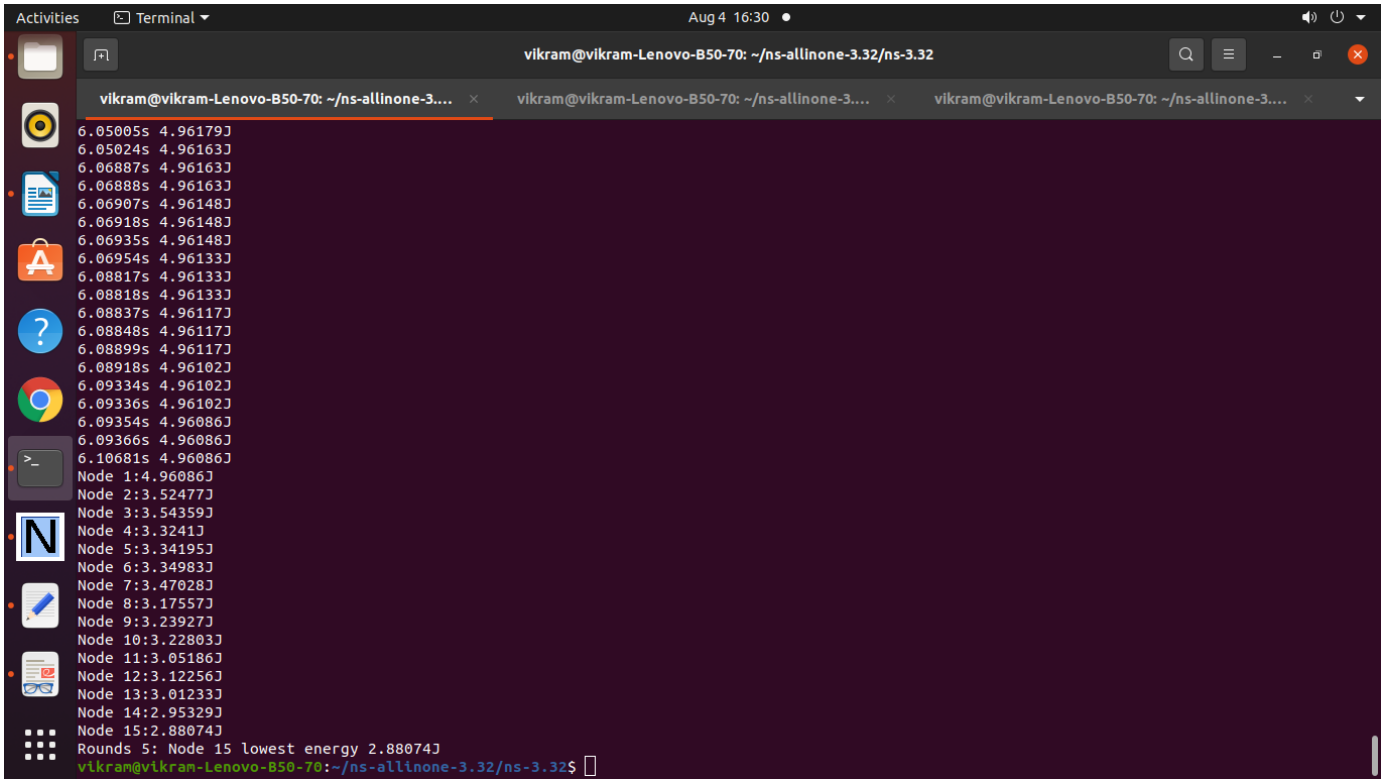
```
void Remain(double old, double newv)
{
  NS_LOG_UNCOND(Simulator::Now().GetSeconds()<<"s "<<newv<<"J");
}

int main(int argc, const char** argv) {

   int Cluster_Size = 15;

   // Create nodes
   Ptr<Node> sink = CreateObject<Node>();
   NodeContainer sensor;
   sensor.Create(Cluster_Size);
   NodeContainer allnode;
   allnode.Add(sink);
   allnode.Add(sensor);

   // set up the phy layer (Wifi model used)
   YansWifiChannelHelper channel;
   channel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel"); //
propagation delay & loss
   channel.AddPropagationLoss("ns3::RangePropagationLossModel",
                        "MaxRange", DoubleValue(35.36));

   YansWifiPhyHelper phy = YansWifiPhyHelper::Default();
   phy.SetChannel(channel.Create()); // error rate

   // mac layer
   WifiHelper wifi;
   wifi.SetStandard(WIFI_STANDARD_80211b);
   wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager",
                           "DataMode", StringValue ("DsssRate1Mbps"),
                           "ControlMode", StringValue ("DsssRate1Mbps"));

   WifiMacHelper mac;
   mac.SetType("ns3::AdhocWifiMac");

   NetDeviceContainer wifide;
   NetDeviceContainer sinkde;
   wifide = wifi.Install(phy, mac, sensor);
   sinkde = wifi.Install(phy, mac, sink);
   NetDeviceContainer allde;
   allde.Add(sinkde);
   allde.Add(wifide);
```

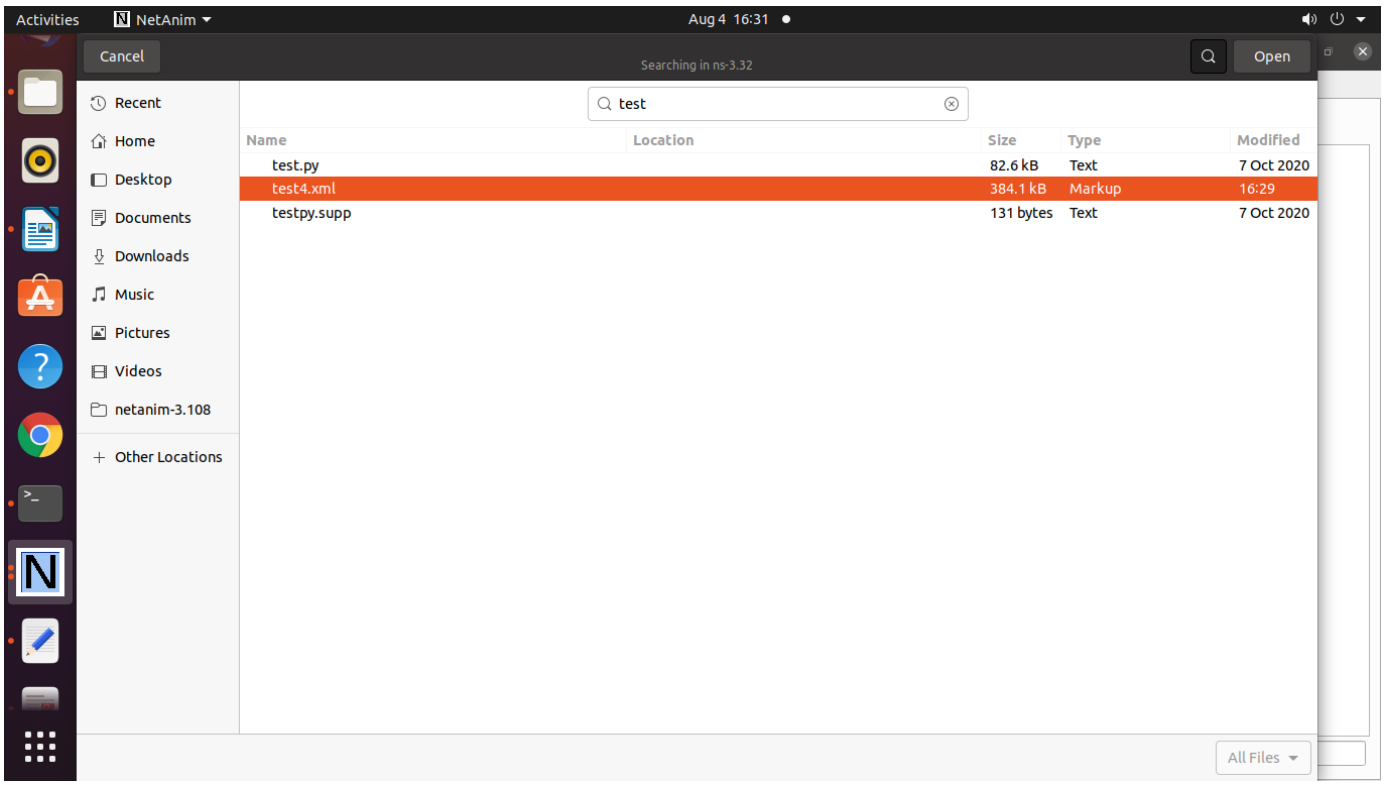
```
   // The location of the nodes
   MobilityHelper model;
   model.SetMobilityModel("ns3::ConstantPositionMobilityModel");
   model.SetPositionAllocator("ns3::RandomRectanglePositionAllocator",
                              "X",
StringValue("ns3::UniformRandomVariable[Min=0|Max=50]"),
                              "Y",
StringValue("ns3::UniformRandomVariable[Min=0|Max=50]"));
   model.Install(sensor);
   model.SetPositionAllocator("ns3::GridPositionAllocator",
                              "MinX", DoubleValue(25),
                              "MinY", DoubleValue(25));
   model.Install(sink);

   // energy module
   BasicEnergySourceHelper energyhelper;
   energyhelper.Set("BasicEnergySourceInitialEnergyJ", DoubleValue(5));
   EnergySourceContainer sources = energyhelper.Install(sensor);

   WifiRadioEnergyModelHelper energymodel;
   energymodel.Set("TxCurrentA", DoubleValue(0.45));
   energymodel.Set("RxCurrentA", DoubleValue(0.21));
   energymodel.Set("IdleCurrentA", DoubleValue(0.1));
   energymodel.Set("SleepCurrentA", DoubleValue(0.03));
   DeviceEnergyModelContainer energydevice = energymodel.Install(wifide,
sources);


   Ptr<BasicEnergySource> sourcePtr =
DynamicCast<BasicEnergySource>(sources.Get(0));
   sourcePtr->TraceConnectWithoutContext("RemainingEnergy",
MakeCallback(&Remain));
   Ptr<DeviceEnergyModel> modelPtr =
sourcePtr->FindDeviceEnergyModels("ns3::WifiRadioEnergyModel").Get(0);

   /** Internet stack **/
   InternetStackHelper internet;
   internet.Install (allnode);

   Ipv4AddressHelper ipv4;
   ipv4.SetBase ("10.1.1.0", "255.255.255.0");
   Ipv4InterfaceContainer il = ipv4.Assign (allde);

   TypeId tid = UdpSocketFactory::GetTypeId();
```

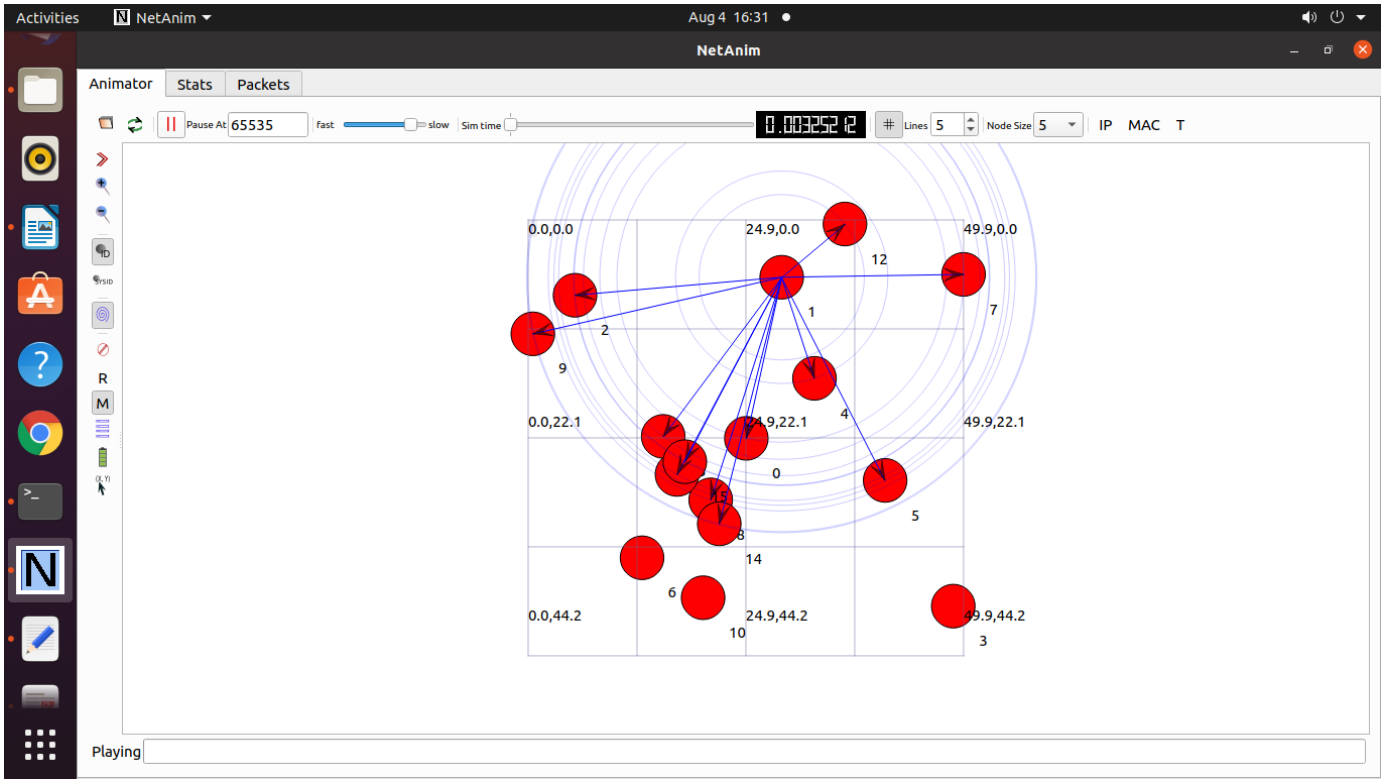
```
   Ptr<Socket> recvSink = Socket::CreateSocket (sink, tid);
   InetSocketAddress local = InetSocketAddress (Ipv4Address(il.GetAddress(0)),
80);
   recvSink->Bind (local);
   recvSink->SetAllowBroadcast(false);

   Ptr<Socket> source[Cluster_Size];

   for(int i=0; i < Cluster_Size; i++){
     InetSocketAddress remote = InetSocketAddress
(Ipv4Address(il.GetAddress(i+1)), 80);
     source[i] = Socket::CreateSocket(sensor.Get(i), tid);
     source[i]->Bind(remote);
     source[i]->SetAllowBroadcast(false);
     source[i]->Connect(local);
   }

   /** simulation setup **/
   AnimationInterface anim ("test4.xml");
   // start traffic
   int rounds = 5;

   // The Send Packet function: In each round, every node will send a packet to sink
node.
   // And it will repeat for given figure, at last, the energyprint function will list the
   // remain energy of each node and tell the one with lowest energy.
   // A real time trace function will tell the time energy stop tracing.
   for (int i = 0; i < rounds; i++)
   {
     SendPacket(source, Cluster_Size, double(2.2*i), modelPtr);
   }

   Simulator::Stop (Seconds (2.2*(rounds+1)));
   Simulator::Run ();

   EnergyPrint(sources, rounds);

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

Output


Activities
Terminal
Aug 4 16:30
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/test2.cc
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.993s)
0.00105s 5J
0.001754s 5J
0.00180812s 5J
0.00199612s 4.99985J
0.00250812s 4.99985J
0.00251812s 4.99985J
0.00282212s 4.99985J
0.00325212s 4.99985J
0.0220681s 4.99985J
0.0220822s 4.99985J
0.0222702s 4.99969J
0.0223822s 4.99969J
0.0227322s 4.99969J
0.0415482s 4.99969J
0.0415624s 4.99969J
0.0417504s 4.99954J
0.0418624s 4.99954J
0.0421524s 4.99954J
0.0465044s 4.99954J
0.0465185s 4.99954J
0.0467065s 4.99938J
0.0468185s 4.99938J
0.156054s 4.99938J
0.156242s 4.99923J
0.156754s 4.99923J
0.156808s 4.99923J
0.156996s 4.99907J
0.157508s 4.99907J
0.157522s 4.99907J
0.15771s 4.99892J
0.157822s 4.99892J


Activities
Terminal
Aug 4 16:30
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
6.05005s 4.96179J
6.05024s 4.96163J
6.06887s 4.96163J
6.06888s 4.96163J
6.06907s 4.96148J
6.06918s 4.96148J
6.06935s 4.96148J
6.06954s 4.96133J
6.08817s 4.96133J
6.08818s 4.96133J
6.08837s 4.96117J
6.08848s 4.96117J
6.08899s 4.96117J
6.08918s 4.96102J
6.09334s 4.96102J
6.09336s 4.96102J
6.09354s 4.96086J
6.09366s 4.96086J
6.10681s 4.96086J
Node 1:4.96086J
Node 2:3.52477J
Node 3:3.54359J
Node 4:3.3241J
Node 5:3.34195J
Node 6:3.34983J
Node 7:3.47028J
Node 8:3.17557J
Node 9:3.23927J
Node 10:3.22803J
Node 11:3.05186J
Node 12:3.12256J
Node 13:3.01233J
Node 14:2.95329J
Node 15:2.88074J
Rounds 5: Node 15 lowest energy 2.88074J
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$


Activities
NetAnim
Aug 4 16:31
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
test
Name
Location
Size
Type
Modified
test.py
82.6 kB
Text
7 Oct 2020
test4.xml
384.1 kB
Markup
16:29
testpy.supp
131 bytes
Text
7 Oct 2020
All Files


Activities
NetAnim
Aug 4 16:31
NetAnim
Animator
Stats
Packets
Pause At
65535
fast
slow
Sim time
Lines
5
Node Size
5
IP
MAC
T
0.0,0.0
24.9,0.0
49.9,0.0
0.0,22.1
24.9,22.1
49.9,22.1
0.0,44.2
24.9,44.2
49.9,44.2
Playing

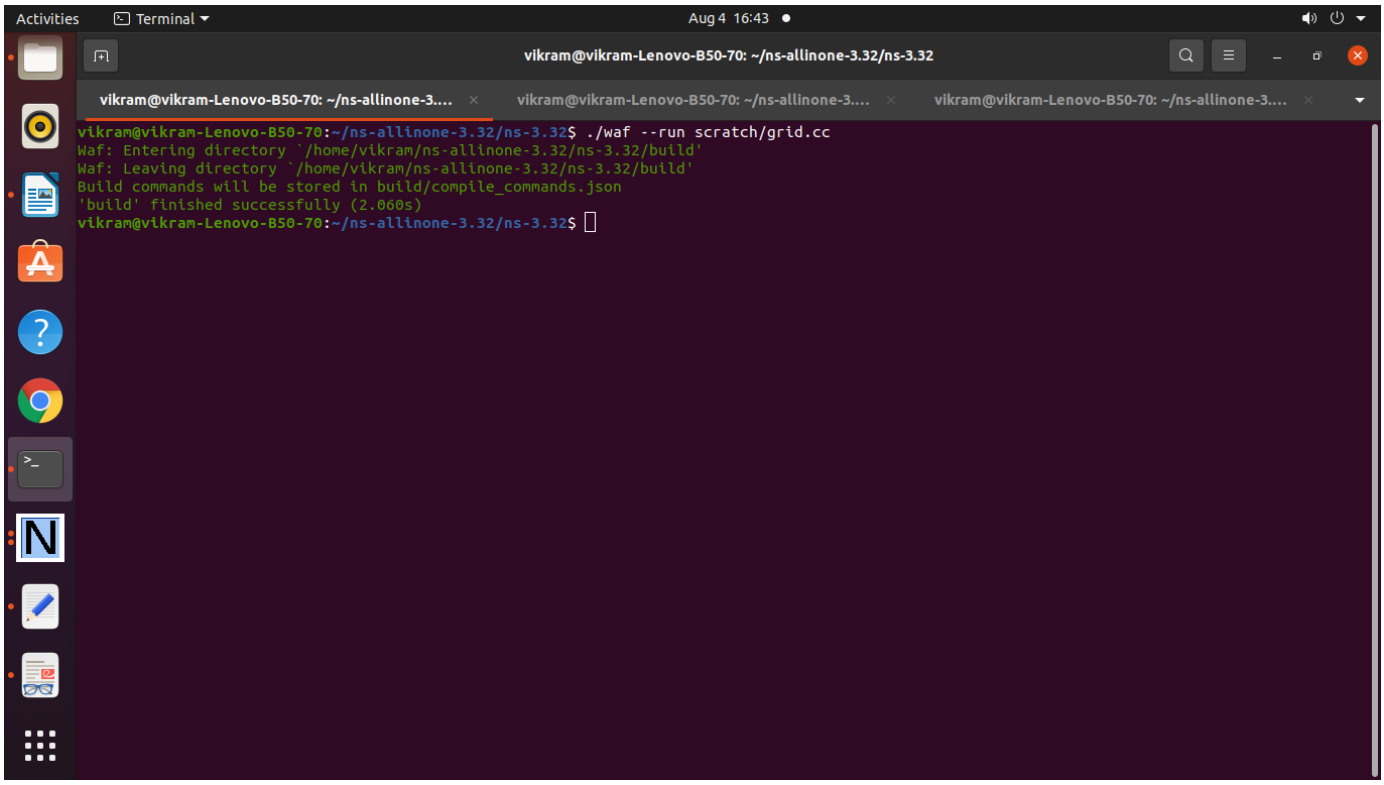
Program 17 : Program to Simulatr Grid-Topology

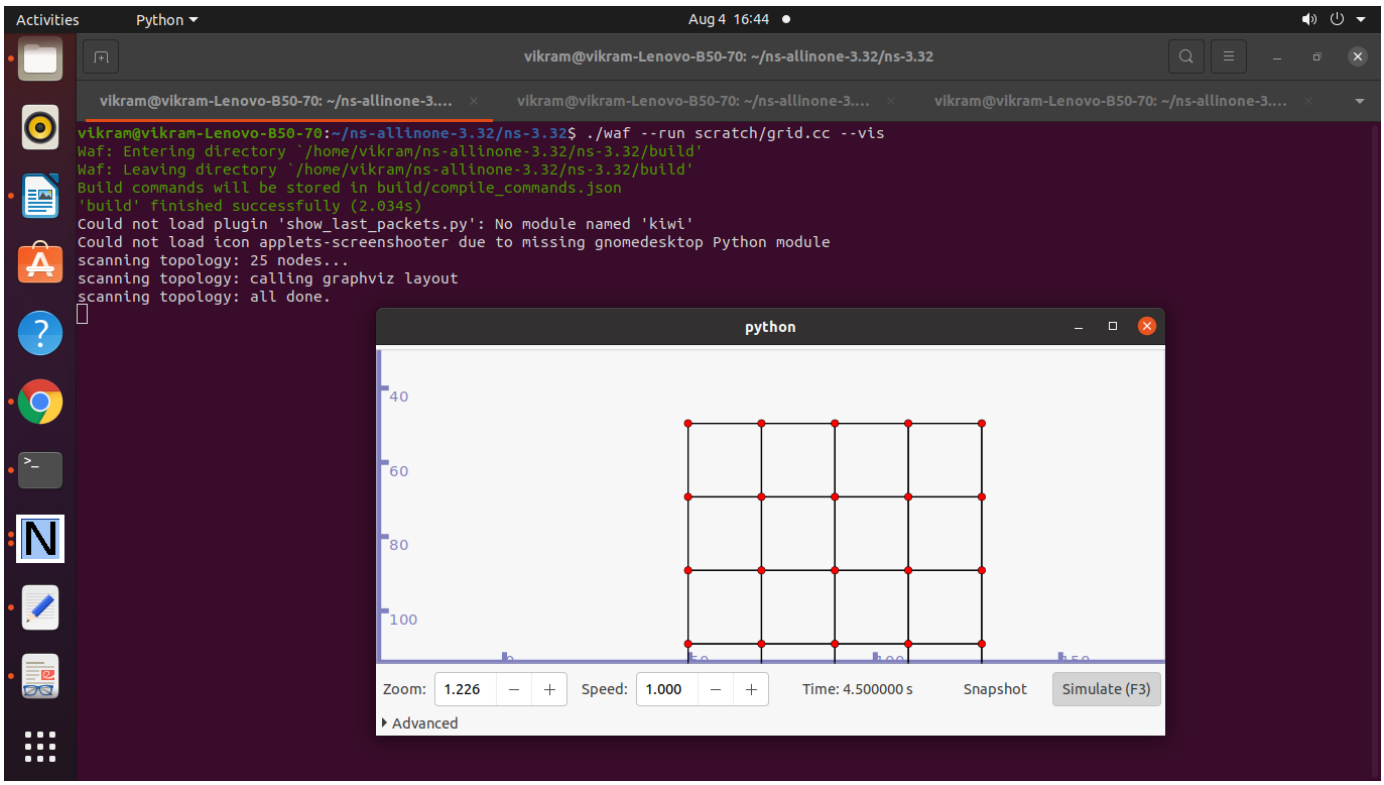
```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
   /*
    * This program is free software; you can redistribute it and/or modify
    * it under the terms of the GNU General Public License version 2 as
    * published by the Free Software Foundation;
    *
   * This program is distributed 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
    *
   * Author: Josh Pelkey <jpelkey@gatech.edu>
   */

   #include <iostream>

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

  using namespace ns3;

  int main (int argc, char *argv[])
  {
    Config::SetDefault ("ns3::OnOffApplication::PacketSize", UintegerValue (512));
    Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue
("500kb/s"));

    uint32_t xSize = 5;
     uint32_t ySize = 5;
     std::string animFile = "grid-animation.xml";

    CommandLine cmd;
    cmd.AddValue ("xSize", "Number of rows of nodes", xSize);
```

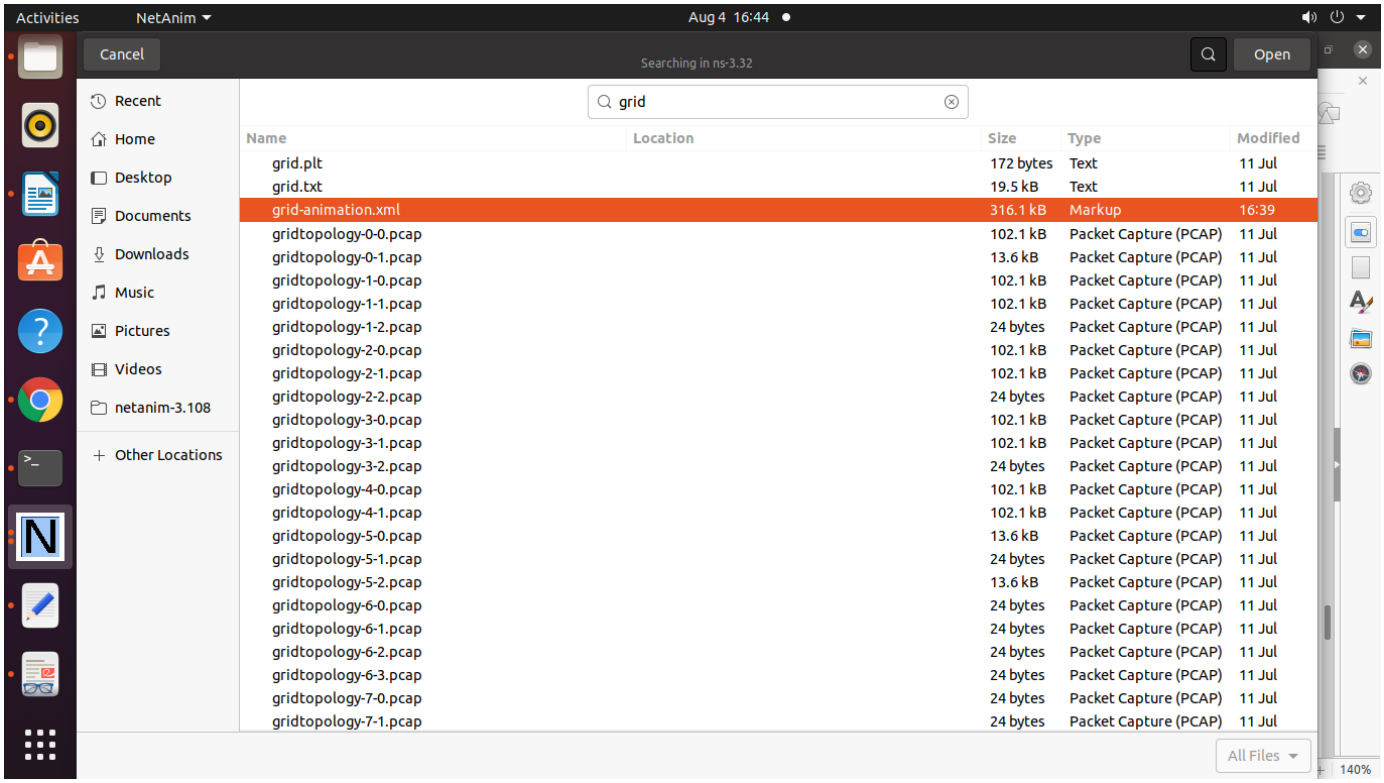
```
    cmd.AddValue ("ySize", "Number of columns of nodes", ySize);
    cmd.AddValue ("animFile",  "File Name for Animation Output", animFile);

    cmd.Parse (argc,argv);
    if (xSize < 1 || ySize < 1 || (xSize < 2 && ySize < 2))
     {
       NS_FATAL_ERROR ("Need more nodes for grid.");
     }

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

     // Create Grid
    PointToPointGridHelper grid (xSize, ySize, pointToPoint);

     // Install stack on Grid
     InternetStackHelper stack;
     grid.InstallStack (stack);

    // Assign Addresses to Grid
    grid.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0", "255.255.255.0"),
                              Ipv4AddressHelper ("10.2.1.0", "255.255.255.0"));


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

     // Create an on/off app sending packets
    AddressValue remoteAddress (InetSocketAddress (grid.GetIpv4Address (xSize-
1,ySize-1), 1000));
    clientHelper.SetAttribute ("Remote", remoteAddress);
     clientApps.Add (clientHelper.Install (grid.GetNode (0,0)));

     clientApps.Start (Seconds (0.0));
     clientApps.Stop (Seconds (1.5));

     // Set the bounding box for animation
     grid.BoundingBox (1, 1, 100, 100);

   // Create the animation object and configure for specified output
```

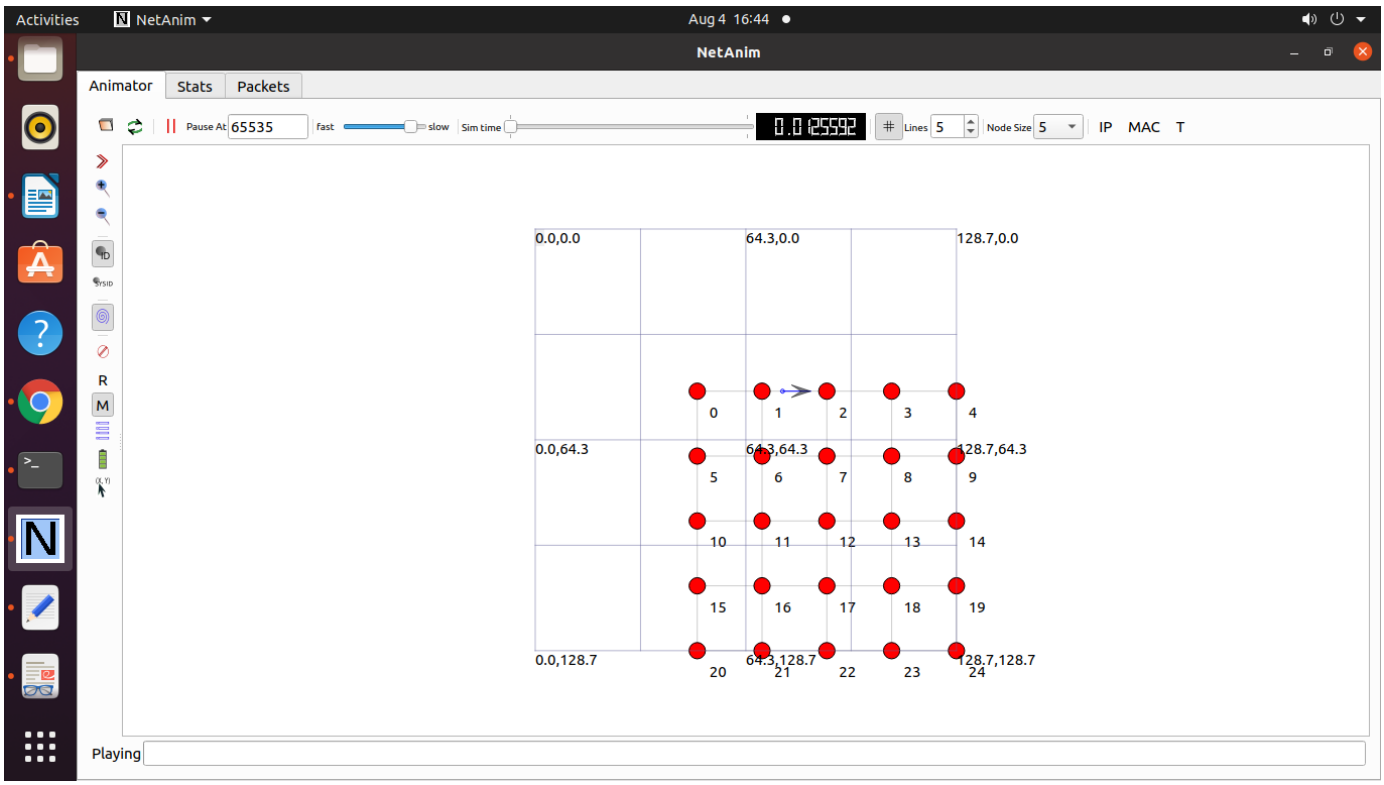
```
  AnimationInterface anim (animFile);

  // Set up the actual simulation
  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

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

Output :



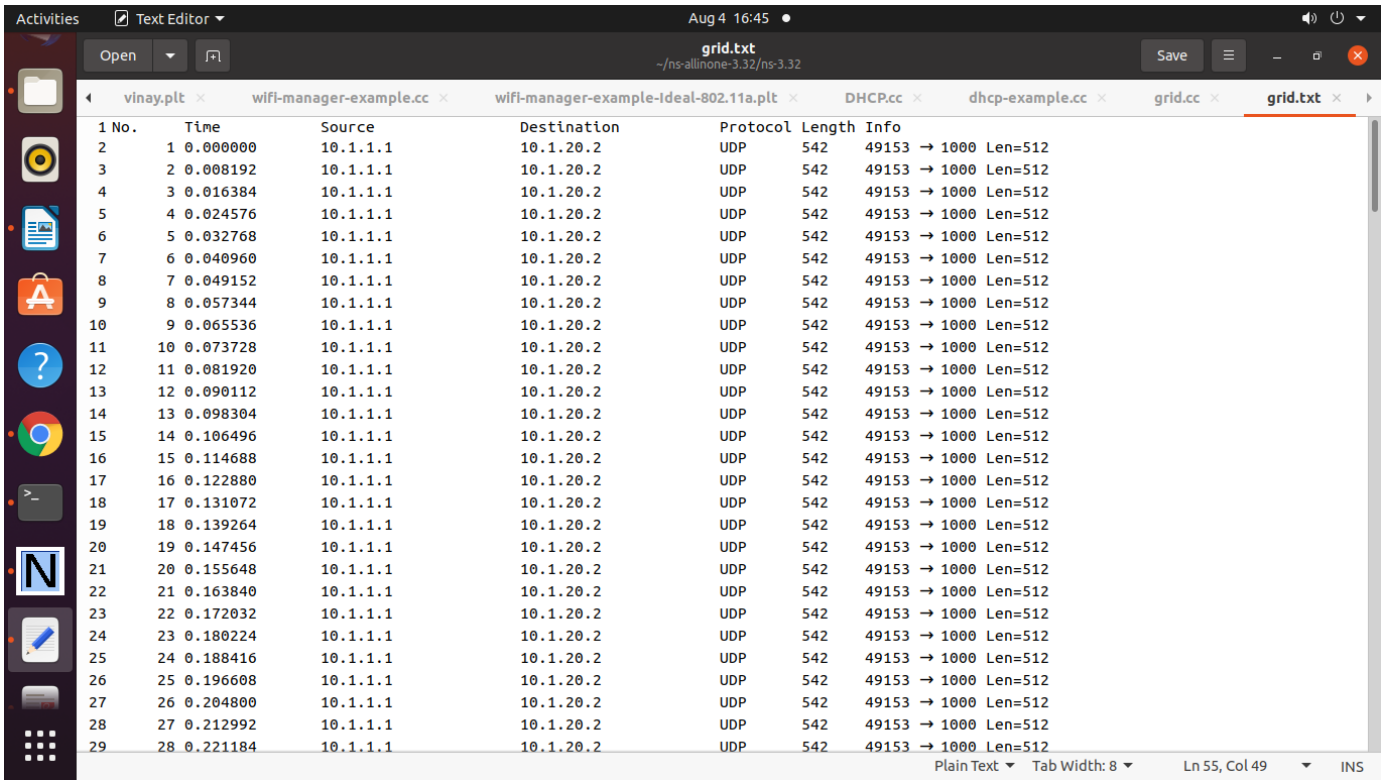

Activities
Python
Aug 4 16:44
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/grid.cc --vis
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.034s)
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: 25 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
python
40
60
80
100
Zoom: 1.226
Speed: 1.000
Time: 4.500000 s
Snapshot
Simulate (F3)
Advanced


Activities
NetAnim
Aug 4 16:44
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
grid
Name
Location
Size
Type
Modified
grid.plt 172 bytes Text 11 Jul
grid.txt 19.5 kB Text 11 Jul
grid-animation.xml 316.1 kB Markup 16:39
gridtopology-0-0.pcap 102.1 kB Packet Capture (PCAP) 11 Jul
gridtopology-0-1.pcap 13.6 kB Packet Capture (PCAP) 11 Jul
gridtopology-1-0.pcap 102.1 kB Packet Capture (PCAP) 11 Jul
gridtopology-1-1.pcap 102.1 kB Packet Capture (PCAP) 11 Jul
gridtopology-1-2.pcap 24 bytes Packet Capture (PCAP) 11 Jul
gridtopology-2-0.pcap 102.1 kB Packet Capture (PCAP) 11 Jul
gridtopology-2-1.pcap 102.1 kB Packet Capture (PCAP) 11 Jul
gridtopology-2-2.pcap 24 bytes Packet Capture (PCAP) 11 Jul
gridtopology-3-0.pcap 102.1 kB Packet Capture (PCAP) 11 Jul
gridtopology-3-1.pcap 102.1 kB Packet Capture (PCAP) 11 Jul
gridtopology-3-2.pcap 24 bytes Packet Capture (PCAP) 11 Jul
gridtopology-4-0.pcap 102.1 kB Packet Capture (PCAP) 11 Jul
gridtopology-4-1.pcap 102.1 kB Packet Capture (PCAP) 11 Jul
gridtopology-5-0.pcap 13.6 kB Packet Capture (PCAP) 11 Jul
gridtopology-5-1.pcap 24 bytes Packet Capture (PCAP) 11 Jul
gridtopology-5-2.pcap 13.6 kB Packet Capture (PCAP) 11 Jul
gridtopology-6-0.pcap 24 bytes Packet Capture (PCAP) 11 Jul
gridtopology-6-1.pcap 24 bytes Packet Capture (PCAP) 11 Jul
gridtopology-6-2.pcap 24 bytes Packet Capture (PCAP) 11 Jul
gridtopology-6-3.pcap 24 bytes Packet Capture (PCAP) 11 Jul
gridtopology-7-0.pcap 24 bytes Packet Capture (PCAP) 11 Jul
gridtopology-7-1.pcap 24 bytes Packet Capture (PCAP) 11 Jul
All Files
140%

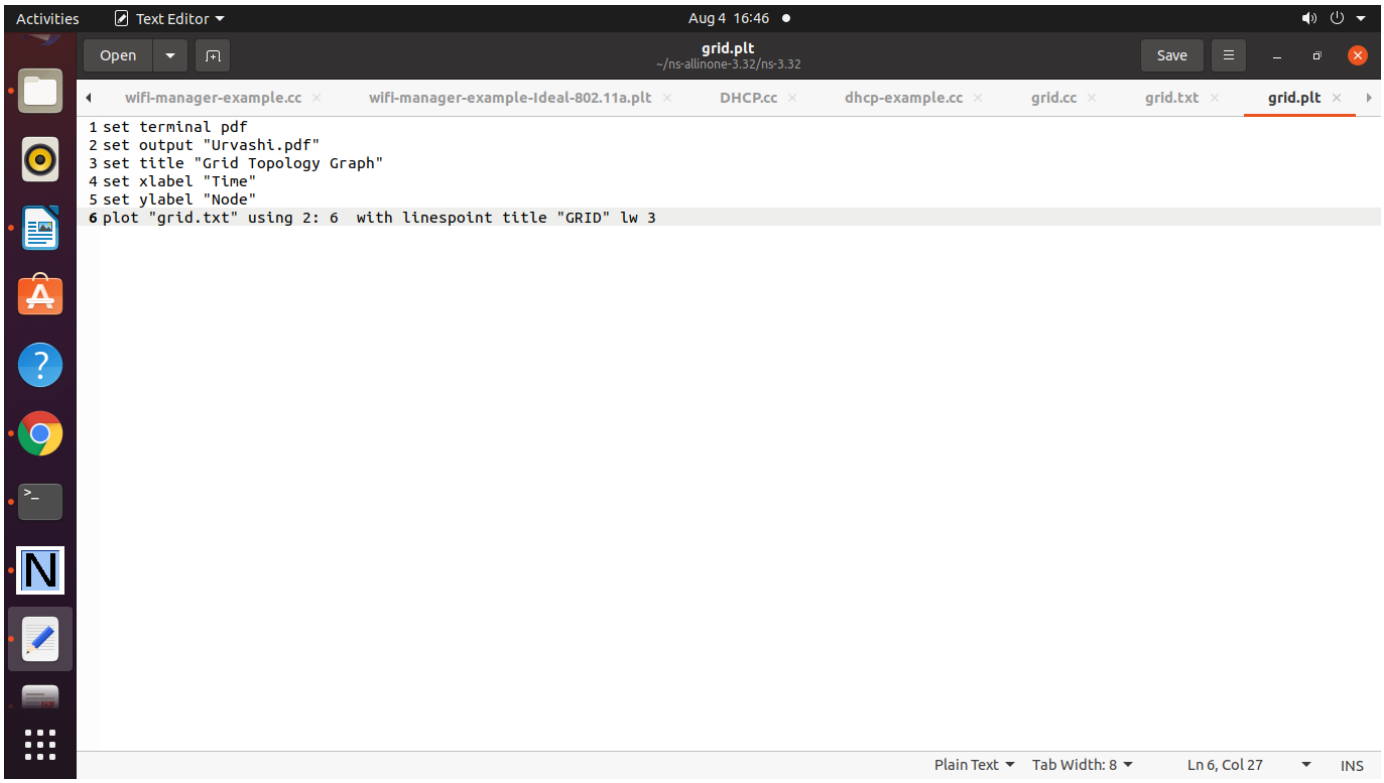


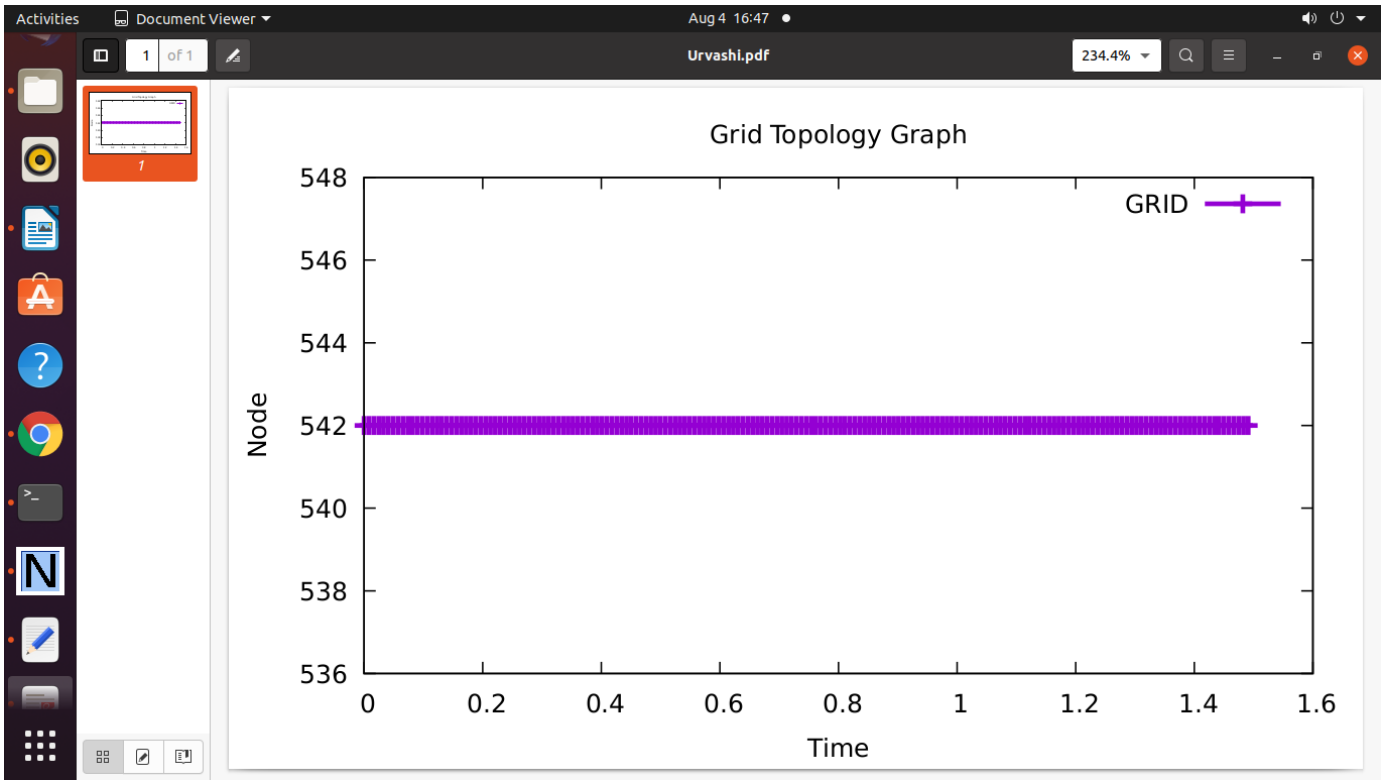
To draw the Graph consider the following files

grid.txt



grid.plt


Activities
Text Editor
Aug 4 16:46
Open
grid.plt
~/ns-allinone-3.32/ns-3.32
Save
wifi-manager-example.cc
wifi-manager-example-Ideal-802.11a.plt
DHCP.cc
dhcp-example.cc
grid.cc
grid.txt
grid.plt
1 set terminal pdf
2 set output "Urvashi.pdf"
3 set title "Grid Topology Graph"
4 set xlabel "Time"
5 set ylabel "Node"
6 plot "grid.txt" using 2: 6 with linespoint title "GRID" lw 3
Plain Text
Tab Width: 8
Ln 6, Col 27
INS


Activities
Document Viewer
Aug 4 16:47
1 of 1
Urvashi.pdf
234.4%
Grid Topology Graph
GRID
Node
548
546
544
542
540
538
536
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
Time

Practical 18 : Program to Simulate Hybird Topology

```
/* -*- 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
 */



/*
        LAB Assignment #2
    1. Create a simple dumbbell topology, two client Node1 and Node2 on
    the left side of the dumbbell and server nodes Node3 and Node4 on the
    right side of the dumbbell. Let Node5 and Node6 form the bridge of the
    dumbbell. Use point to point links.

    2. Install a TCP socket instance on Node1 that will connect to Node3.

    3. Install a UDP socket instance on Node2 that will connect to Node4.

    4. Start the TCP application at time 1s.

    5. Start the UDP application at time 20s at rate Rate1 such that it clogs
    half the dumbbell bridge's link capacity.

    6. Increase the UDP application's rate at time 30s to rate Rate2
    such that it clogs the whole of the dumbbell bridge's capacity.

    7. Use the ns-3 tracing mechanism to record changes in congestion window
    size of the TCP instance over time. Use gnuplot/matplotlib to visualise plots of
cwnd vs time.
```

8. Mark points of fast recovery and slow start in the graphs.

9. Perform the above experiment for TCP variants Tahoe, Reno and New Reno, all of which are available with ns-3.

Solution by: Konstantinos Katsaros (K.Katsaros@surrey.ac.uk)
based on fifth.cc
*/

```
// Network topology
//
//       n0 ---+     +--- n2
//             |     |
//            n4 -- n5
//             |     |
//       n1 ---+     +--- n3
//
// - All links are P2P with 500kb/s and 2ms
// - TCP flow form n0 to n2
// - UDP flow from n1 to n3

#include <fstream>
#include <string>
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/internet-module.h"
#include "ns3/flow-monitor-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 ("Lab2");

class MyApp : public Application
{
public:

  MyApp ();
  virtual ~MyApp();

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

```
  void ChangeRate(DataRate newrate);

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)
    {
```

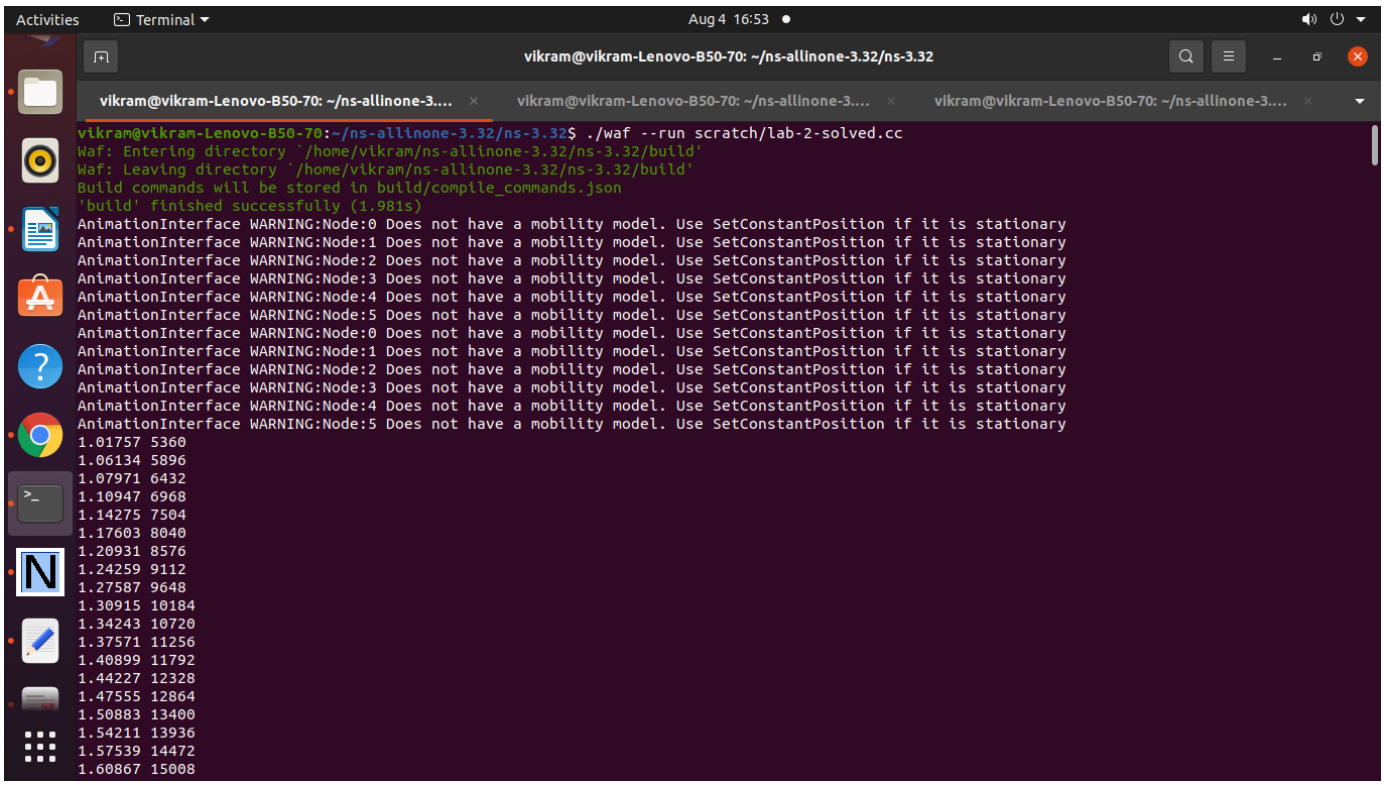
```
    Time tNext (Seconds (m_packetSize * 8 / static_cast<double>
(m_dataRate.GetBitRate ())));
    m_sendEvent = Simulator::Schedule (tNext, &MyApp::SendPacket, this);
   }
}

void
MyApp::ChangeRate(DataRate newrate)
{
  m_dataRate = newrate;
  return;
}

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

void
IncRate (Ptr<MyApp> app, DataRate rate)
{
        app->ChangeRate(rate);
   return;
}


int main (int argc, char *argv[])
{
 std::string lat = "2ms";
 std::string rate = "500kb/s"; // P2P link
 bool enableFlowMonitor = false;


 CommandLine cmd;
 cmd.AddValue ("latency", "P2P link Latency in miliseconds", lat);
 cmd.AddValue ("rate", "P2P data rate in bps", rate);
 cmd.AddValue ("EnableMonitor", "Enable Flow Monitor", enableFlowMonitor);

 cmd.Parse (argc, argv);

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

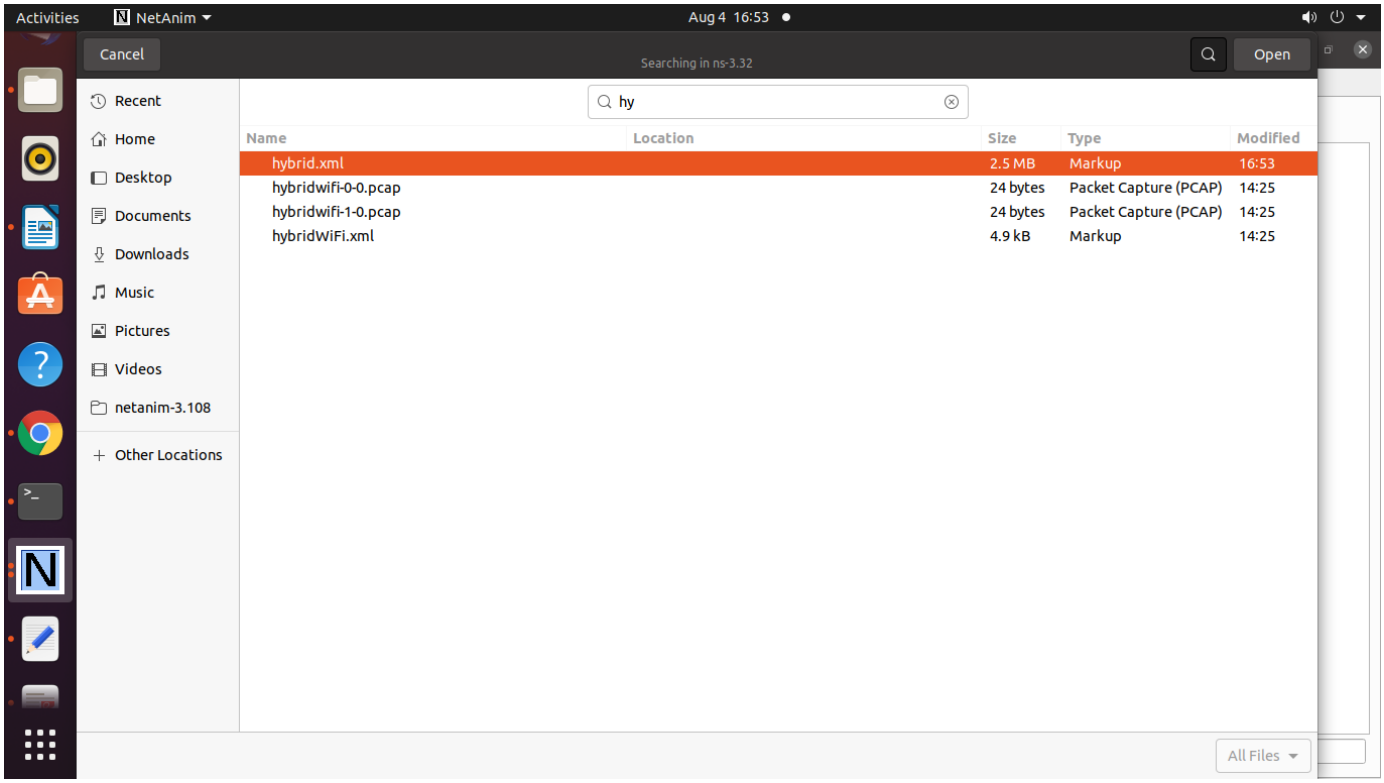
```
  NodeContainer c; // ALL Nodes
  c.Create(6);

  NodeContainer n0n4 = NodeContainer (c.Get (0), c.Get (4));
  NodeContainer n1n4 = NodeContainer (c.Get (1), c.Get (4));
  NodeContainer n2n5 = NodeContainer (c.Get (2), c.Get (5));
  NodeContainer n3n5 = NodeContainer (c.Get (3), c.Get (5));
  NodeContainer n4n5 = NodeContainer (c.Get (4), c.Get (5));

//
// Install Internet Stack
//
  InternetStackHelper internet;
  internet.Install (c);

  // We create the channels first without any IP addressing information
  NS_LOG_INFO ("Create channels.");
  PointToPointHelper p2p;
  p2p.SetDeviceAttribute ("DataRate", StringValue (rate));
  p2p.SetChannelAttribute ("Delay", StringValue (lat));
  NetDeviceContainer d0d4 = p2p.Install (n0n4);
  NetDeviceContainer d1d4 = p2p.Install (n1n4);
  NetDeviceContainer d4d5 = p2p.Install (n4n5);
  NetDeviceContainer d2d5 = p2p.Install (n2n5);
  NetDeviceContainer d3d5 = p2p.Install (n3n5);

    // Later, we add IP addresses.
  NS_LOG_INFO ("Assign IP Addresses.");
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer i0i4 = ipv4.Assign (d0d4);

  ipv4.SetBase ("10.1.2.0", "255.255.255.0");
  Ipv4InterfaceContainer i1i4 = ipv4.Assign (d1d4);

  ipv4.SetBase ("10.1.3.0", "255.255.255.0");
  Ipv4InterfaceContainer i4i5 = ipv4.Assign (d4d5);

  ipv4.SetBase ("10.1.4.0", "255.255.255.0");
  Ipv4InterfaceContainer i2i5 = ipv4.Assign (d2d5);

  ipv4.SetBase ("10.1.5.0", "255.255.255.0");
  Ipv4InterfaceContainer i3i5 = ipv4.Assign (d3d5);

  NS_LOG_INFO ("Enable static global routing.");
```

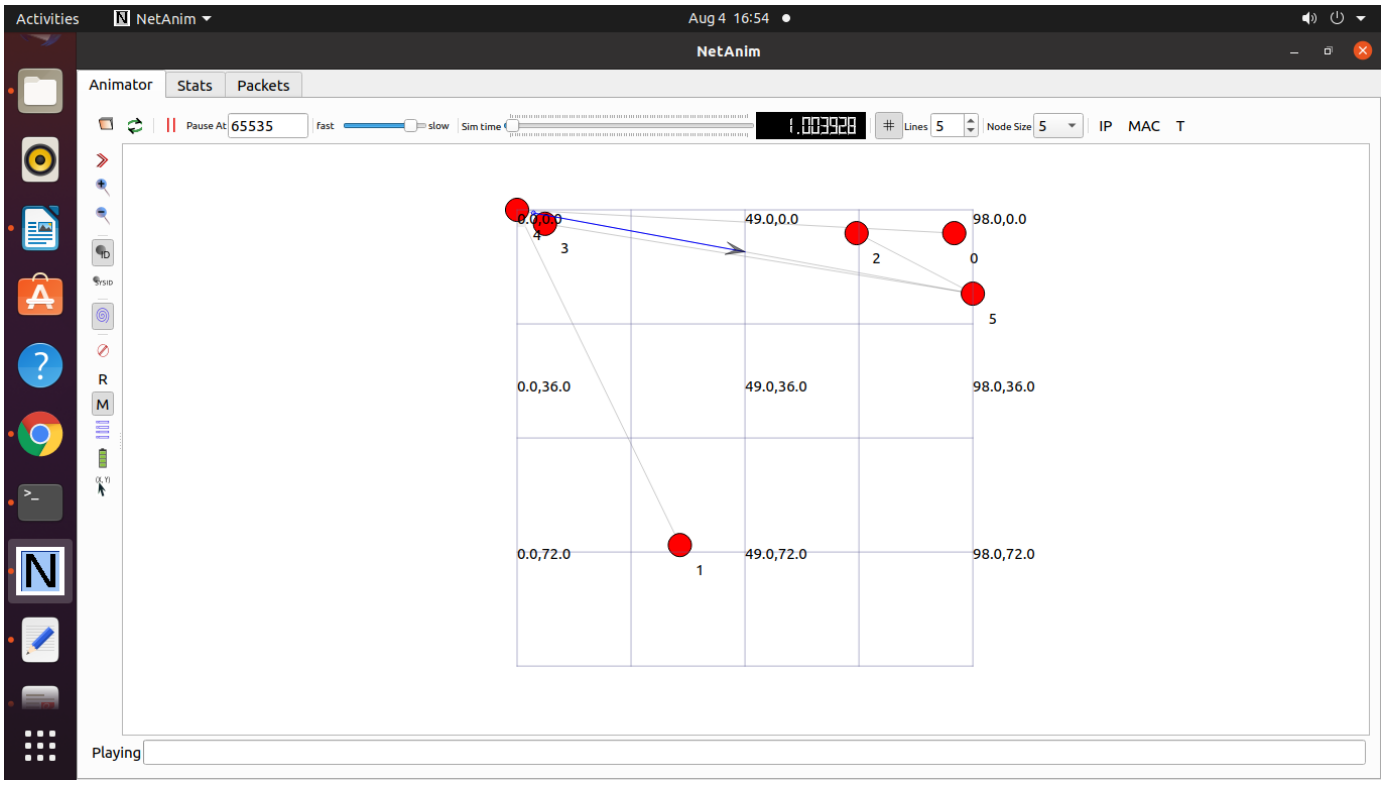
```
  //
  // Turn on global static routing so we can actually be routed across the network.
  //
  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();


  NS_LOG_INFO ("Create Applications.");

  // TCP connfection from N0 to N2

  uint16_t sinkPort = 8080;
  Address sinkAddress (InetSocketAddress (i2i5.GetAddress (0), sinkPort)); //
interface of n2
  PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory",
InetSocketAddress (Ipv4Address::GetAny (), sinkPort));
  ApplicationContainer sinkApps = packetSinkHelper.Install (c.Get (2)); //n2 as
sink
  sinkApps.Start (Seconds (0.));
  sinkApps.Stop (Seconds (100.));

  Ptr<Socket> ns3TcpSocket = Socket::CreateSocket (c.Get (0),
TcpSocketFactory::GetTypeId ()); //source at n0

  // Trace Congestion window
  ns3TcpSocket->TraceConnectWithoutContext ("CongestionWindow",
MakeCallback (&CwndChange));

  // Create TCP application at n0
  Ptr<MyApp> app = CreateObject<MyApp> ();
  app->Setup (ns3TcpSocket, sinkAddress, 1040, 100000, DataRate ("250Kbps"));
  c.Get (0)->AddApplication (app);
  app->SetStartTime (Seconds (1.));
  app->SetStopTime (Seconds (100.));


  // UDP connfection from N1 to N3

  uint16_t sinkPort2 = 6;
  Address sinkAddress2 (InetSocketAddress (i3i5.GetAddress (0), sinkPort2)); //
interface of n3
  PacketSinkHelper packetSinkHelper2 ("ns3::UdpSocketFactory",
InetSocketAddress (Ipv4Address::GetAny (), sinkPort2));
  ApplicationContainer sinkApps2 = packetSinkHelper2.Install (c.Get (3)); //n3 as
sink
  sinkApps2.Start (Seconds (0.));
```

```
  sinkApps2.Stop (Seconds (100.));

  Ptr<Socket> ns3UdpSocket = Socket::CreateSocket (c.Get (1),
UdpSocketFactory::GetTypeId ()); //source at n1

  // Create UDP application at n1
  Ptr<MyApp> app2 = CreateObject<MyApp> ();
  app2->Setup (ns3UdpSocket, sinkAddress2, 1040, 100000, DataRate
("250Kbps"));
  c.Get (1)->AddApplication (app2);
  app2->SetStartTime (Seconds (20.));
  app2->SetStopTime (Seconds (100.));

// Increase UDP Rate
  Simulator::Schedule (Seconds(30.0), &IncRate, app2, DataRate("500kbps"));

  // Flow Monitor
  Ptr<FlowMonitor> flowmon;
  if (enableFlowMonitor)
    {
     FlowMonitorHelper flowmonHelper;
     flowmon = flowmonHelper.InstallAll ();
    }
    AnimationInterface anim("hybrid.xml");

//
// Now, do the actual simulation.
//
  NS_LOG_INFO ("Run Simulation.");
  Simulator::Stop (Seconds(100.0));
  Simulator::Run ();
  if (enableFlowMonitor)
    {
          flowmon->CheckForLostPackets ();
          flowmon->SerializeToXmlFile("lab-2.flowmon", true, true);
    }
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");
}
```

Output


Activities
Terminal
Aug 4 16:53
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/lab-2-solved.cc
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.981s)
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:5 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:5 Does not have a mobility model. Use SetConstantPosition if it is stationary
1.01757 5360
1.06134 5896
1.07971 6432
1.10947 6968
1.14275 7504
1.17603 8040
1.20931 8576
1.24259 9112
1.27587 9648
1.30915 10184
1.34243 10720
1.37571 11256
1.40899 11792
1.44227 12328
1.47555 12864
1.50883 13400
1.54211 13936
1.57539 14472
1.60867 15008


Activities
NetAnim
Aug 4 16:53
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
hy
Name
Location
Size
Type
Modified
hybrid.xml
2.5 MB
Markup
16:53
hybridwifi-0-0.pcap
24 bytes
Packet Capture (PCAP)
14:25
hybridwifi-1-0.pcap
24 bytes
Packet Capture (PCAP)
14:25
hybridWiFi.xml
4.9 kB
Markup
14:25
All Files


Activities
NetAnim
Aug 4 16:54
NetAnim
Animator
Stats
Packets
Pause At
65535
fast
slow
Sim time
Lines
5
Node Size
5
IP
MAC
T
0.0,0.0
49.0,0.0
98.0,0.0
0.0,36.0
49.0,36.0
98.0,36.0
0.0,72.0
49.0,72.0
98.0,72.0
4
3
2
0
5
1
Playing

Program 20 : Program to simulatr Mesh Topology

```
/* -*- 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"
#include "ns3/netanim-module.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")
{
}
```

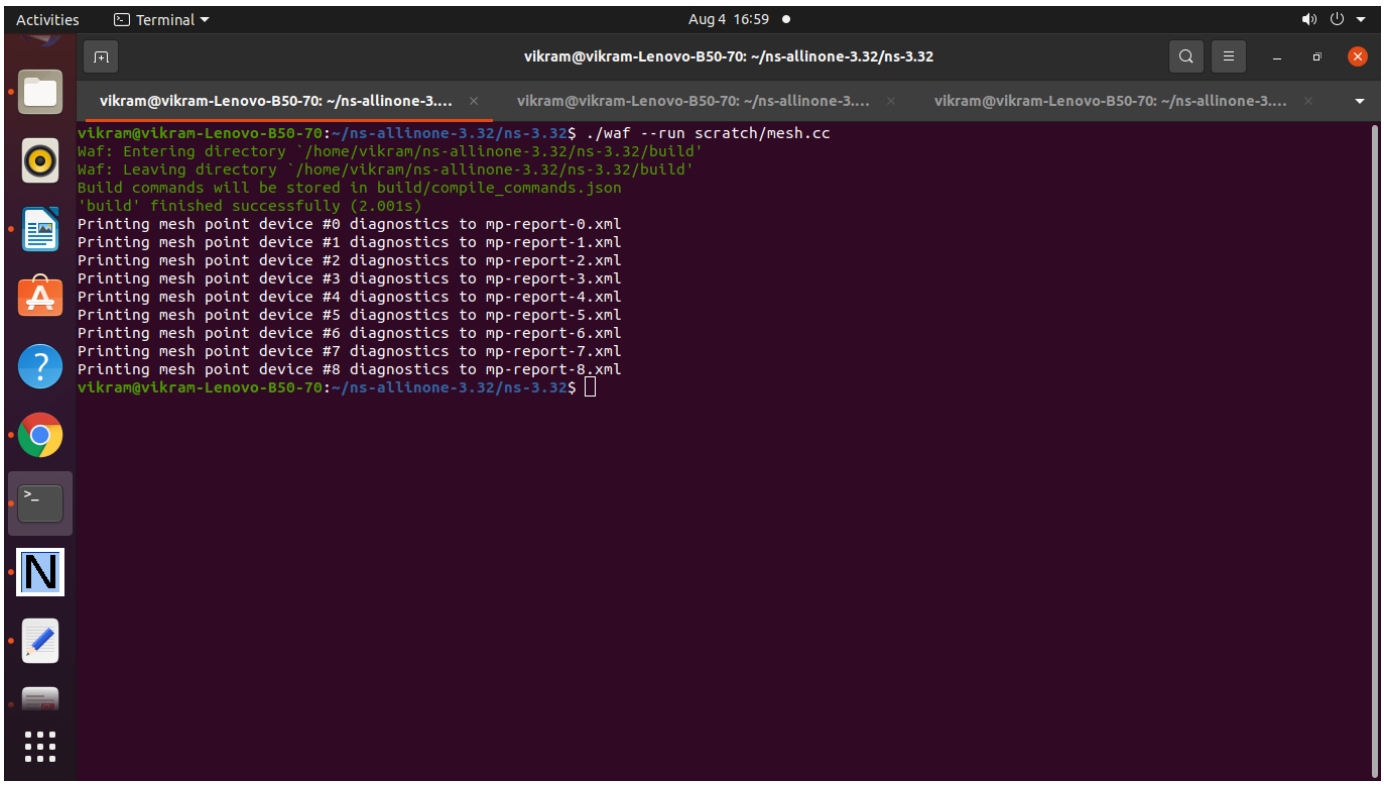
```
void
MeshTest::Configure (int argc, char *argv[])
{
  CommandLine cmd (__FILE__);
  cmd.AddValue ("x-size", "Number of nodes in a row grid", m_xSize);
  cmd.AddValue ("y-size", "Number of rows in a grid", m_ySize);
  cmd.AddValue ("step",   "Size of edge in our grid (meters)", m_step);
  // Avoid 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 = YansWifiPhyHelper::Default ();
  YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
  wifiPhy.SetChannel (wifiChannel.Create ());
  /*
```

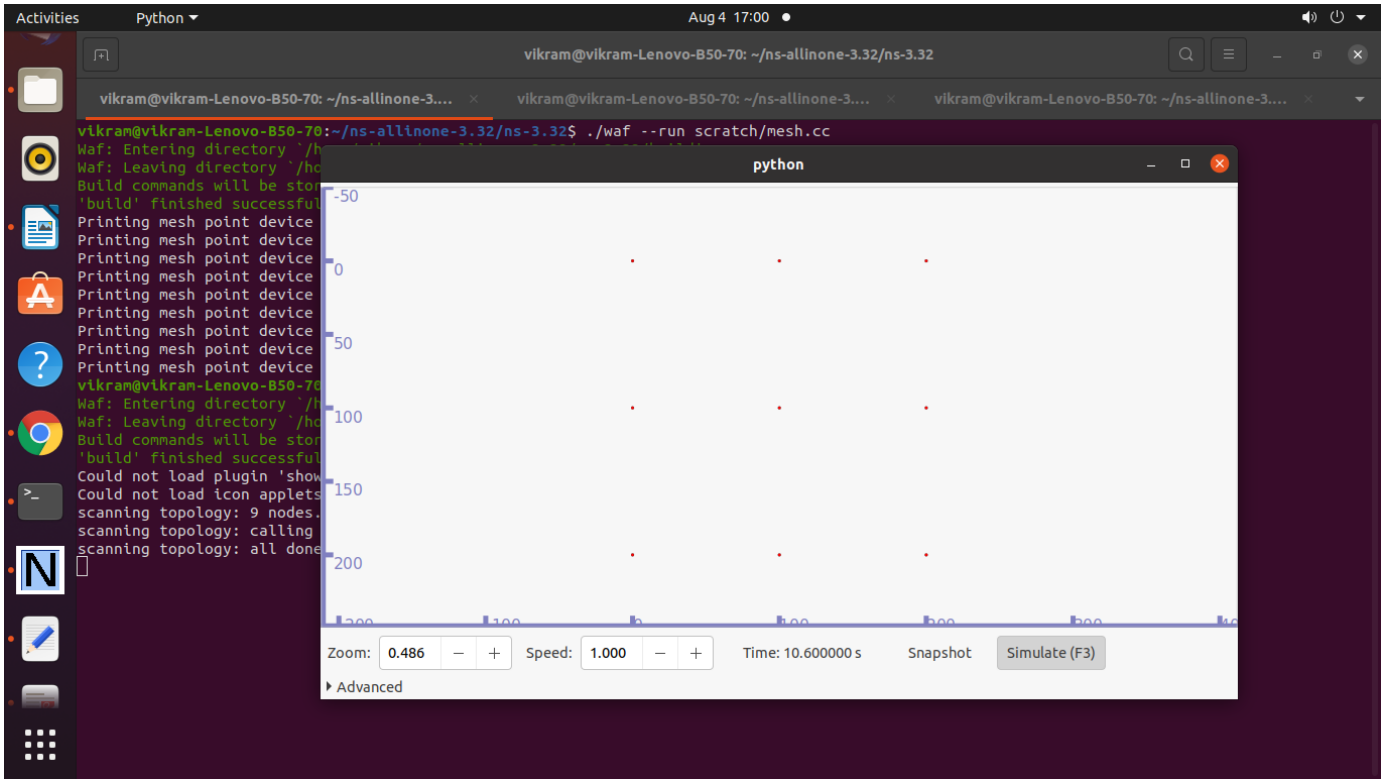
```
   * Create mesh helper and set stack installer to it
   * Stack installer creates all needed protocols and install them to
   * mesh point device
   */
  mesh = MeshHelper::Default ();
  if (!Mac48Address (m_root.c_str ()).IsBroadcast ())
    {
      mesh.SetStackInstaller (m_stack, "Root", Mac48AddressValue (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;
```

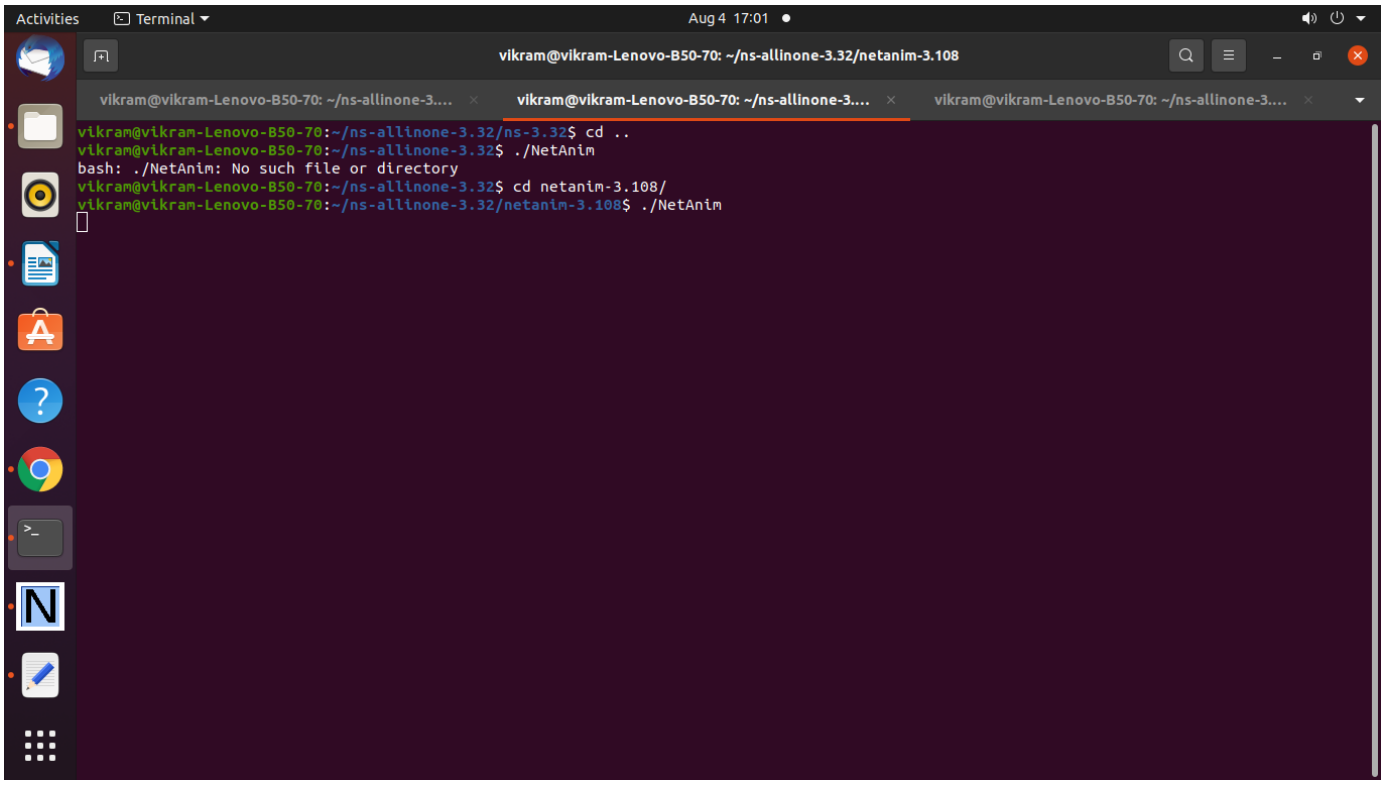
```
    wifiPhy.EnableAsciiAll (ascii.CreateFileStream ("mesh.tr"));
   }
}
void
MeshTest::InstallInternetStack ()
{
 InternetStackHelper internetStack;
 internetStack.Install (nodes);
 Ipv4AddressHelper address;
 address.SetBase ("10.1.1.0", "255.255.255.0");
 interfaces = address.Assign (meshDevices);
}
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));
 AnimationInterface anim("meshdemo.xml");
 Simulator::Run ();
 Simulator::Destroy ();
 return 0;
}
void
MeshTest::Report ()
{
```

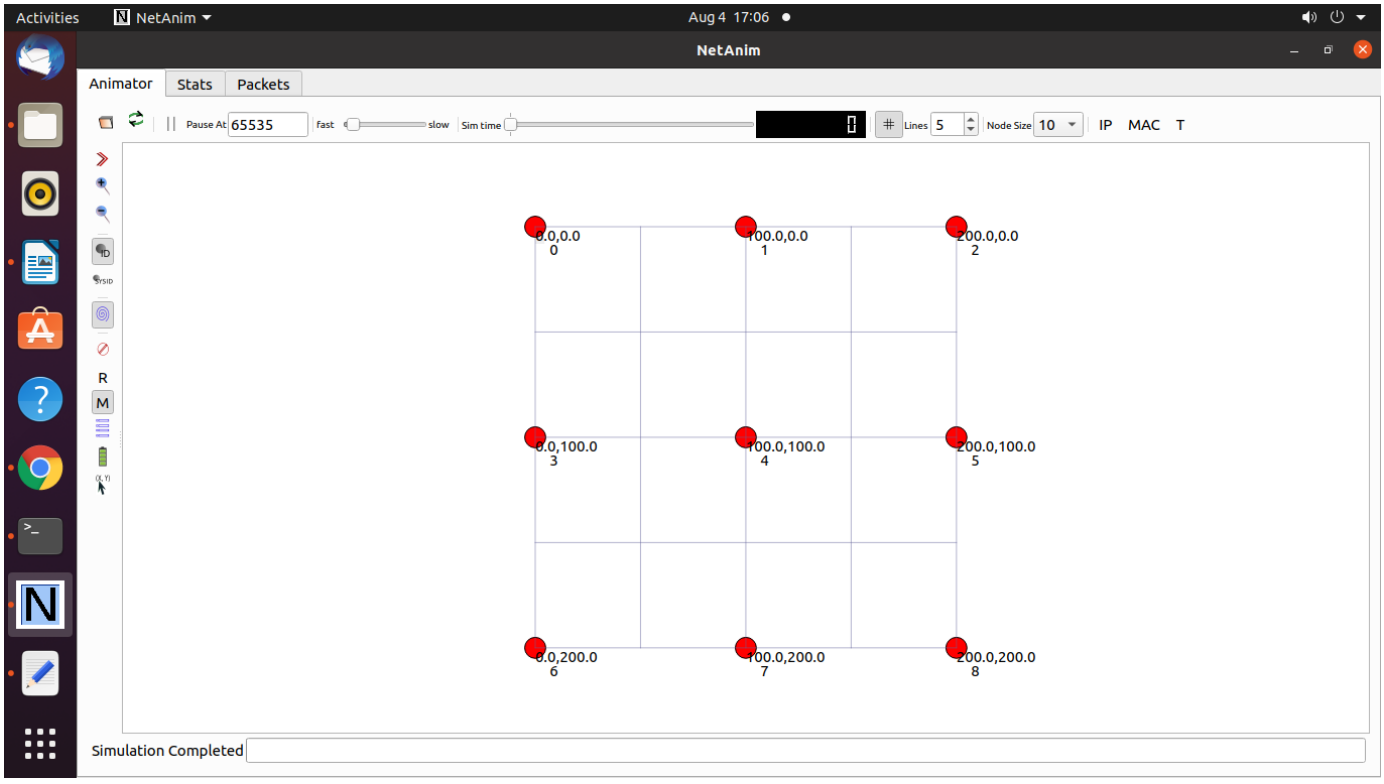
```
  unsigned n (0);
  for (NetDeviceContainer::Iterator i = meshDevices.Begin (); i != meshDevices.End
();  ++i, ++n)
    {
      std::ostringstream os;
      os << "mp-report-" << n << ".xml";
      std::cerr << "Printing mesh point device #" << n << " diagnostics to " << os.str
() << "\n";
      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 ();
}
```

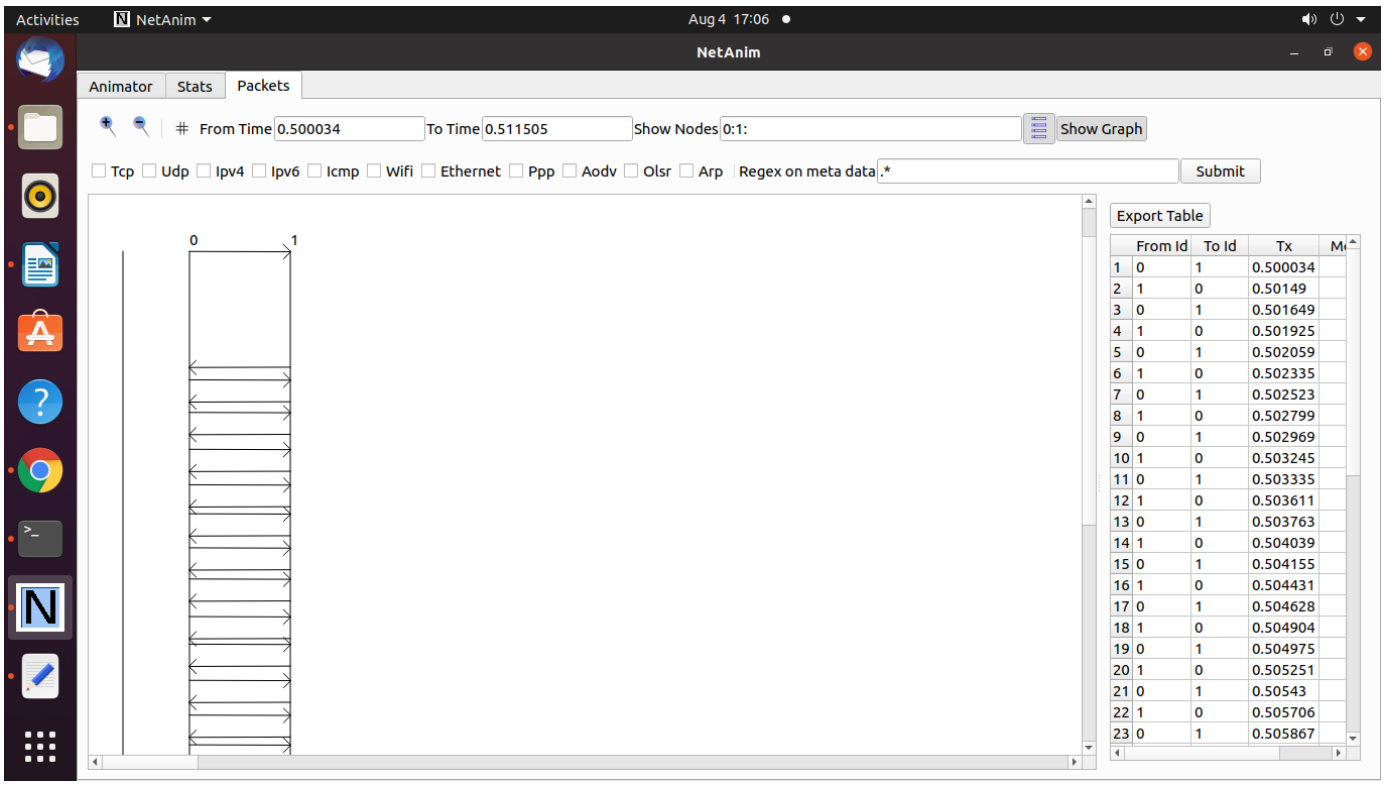
Output:


Activities
Terminal
Aug 4 16:59
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/mesh.cc
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.001s)
Printing mesh point device #0 diagnostics to mp-report-0.xml
Printing mesh point device #1 diagnostics to mp-report-1.xml
Printing mesh point device #2 diagnostics to mp-report-2.xml
Printing mesh point device #3 diagnostics to mp-report-3.xml
Printing mesh point device #4 diagnostics to mp-report-4.xml
Printing mesh point device #5 diagnostics to mp-report-5.xml
Printing mesh point device #6 diagnostics to mp-report-6.xml
Printing mesh point device #7 diagnostics to mp-report-7.xml
Printing mesh point device #8 diagnostics to mp-report-8.xml
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$


Activities
Python
Aug 4 17:00
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/mesh.cc
python
-50
0
50
100
150
200
Zoom: 0.486
Speed: 1.000
Time: 10.600000 s
Snapshot
Simulate (F3)
Advanced


Activities
Terminal
Aug 4 17:01
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/netanim-3.108
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ cd ..
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32$ ./NetAnim
bash: ./NetAnim: No such file or directory
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32$ cd netanim-3.108/
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/netanim-3.108$ ./NetAnim


Activities
NetAnim
Aug 4 17:06
NetAnim
Animator
Stats
Packets
Pause At
65535
fast
slow
Sim time
Lines
5
Node Size
10
IP
MAC
T
0.0,0.0
0
100.0,0.0
1
200.0,0.0
2
0.0,100.0
3
100.0,100.0
4
200.0,100.0
5
0.0,200.0
6
100.0,200.0
7
200.0,200.0
8
Simulation Completed


Activities
NetAnim
Aug 4 17:06
NetAnim
Animator
Stats
Packets
From Time 0.500034
To Time 0.511505
Show Nodes 0:1:
Show Graph
Tcp Udp Ipv4 Ipv6 Icmp Wifi Ethernet Ppp Aodv Olsr Arp
Regex on meta data .*
Submit
0
1
Export Table
From Id To Id Tx
1 0 1 0.500034
2 1 0 0.50149
3 0 1 0.501649
4 1 0 0.501925
5 0 1 0.502059
6 1 0 0.502335
7 0 1 0.502523
8 1 0 0.502799
9 0 1 0.502969
10 1 0 0.503245
11 0 1 0.503335
12 1 0 0.503611
13 0 1 0.503763
14 1 0 0.504039
15 0 1 0.504155
16 1 0 0.504431
17 0 1 0.504628
18 1 0 0.504904
19 0 1 0.504975
20 1 0 0.505251
21 0 1 0.50543
22 1 0 0.505706
23 0 1 0.505867

Practical 21 : Program to Simulate handover in Mesh Topology

```
/* Net topology
 *
 *
 *
 * With a 3x3 mesh grid (size can vary):
 *
 *            n2   n3   n4
 *
 *
 * n0----n1 )))  n5   n6   n7
 *
 *
 *            n8   n9   n10
 *
 *  n0: Internet node
 *  n1: access node (a static mesh node with a P2P link)
 *  n2-n10: mesh nodes
 *
 */



#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mesh-module.h"
#include "ns3/mobility-module.h"
#include "ns3/mesh-helper.h"
#include "ns3/point-to-point-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"
#include "ns3/olsr-helper.h"
#include "ns3/v4ping-helper.h"
#include "ns3/olsr-helper.h"
#include "ns3/aodv-helper.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "src/core/model/string.h"
#include "myapp.h"

#include <iostream>
```

```
#include <sstream>
#include <fstream>
// Needed for ping and test
#include "ns3/test.h"
#include "ns3/abort.h"
#include "ns3/v4ping.h"
#include "ns3/config.h"
#include "ns3/names.h"
#include "src/core/model/log.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("iMesh-handover");


static void
SetPosition (Ptr<Node> node, double x, double y)
{
  Ptr<MobilityModel> mobility = node->GetObject<MobilityModel> ();
  Vector pos = mobility->GetPosition();
  pos.x = x;
  pos.y = y;
  mobility->SetPosition(pos);
}


class MeshTest
{
public:

  /// Init test
  MeshTest ();
  /// Configure test from command line arguments
  void Configure (int argc, char ** argv);
  /// Run test
  int Run ();
private:
  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;
  std::string m_stack;
  std::string m_root;

  NodeContainer nc_all; // Contains every node (starting with internet node, access
nodes and then mesh nodes
  NodeContainer nc_mesh; // Contains mesh nodes
  NodeContainer nc_wireless; // Contains access and mesh nodes
  NodeContainer nc_internet; // Contains internet nodes
  NodeContainer nc_destination; // Contains nodes with sink for apps (internet
and mesh nodes)

  /// List of all mesh point devices
  NetDeviceContainer meshDevices;
  NetDeviceContainer internetDevices;

  //Addresses of meshInterfaces:
  Ipv4InterfaceContainer meshInterfaces;
  Ipv4InterfaceContainer p2pInterfaces; /// Needed for n0-----n1

  // MeshHelper. Report is not static methods
  MeshHelper meshHelper;
  PointToPointHelper p2pHelper;

  MobilityHelper mobility;

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 (30.0),
m_packetInterval (1.0),
m_packetSize (1024),
m_nIfaces (1),
m_chan (true),
m_pcap (true),
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", "Number of nodes in a row grid. [6]", m_xSize);
  cmd.AddValue ("y-size", "Number of rows in a grid. [6]", m_ySize);
  cmd.AddValue ("step", "Size of edge in our grid, meters. [100 m]", m_step);
  /*
   * As soon as starting node means that it sends a beacon,
   * simultaneous start is not good.
   */
  cmd.AddValue ("start", "Maximum random start delay, seconds. [0.1 s]",
m_randomStart);
  cmd.AddValue ("time", "Simulation time, seconds [100 s]", m_totalTime);
  cmd.AddValue ("packet-interval", "Interval between packets in UDP ping, seconds
[0.001 s]", m_packetInterval);
  cmd.AddValue ("packet-size", "Size of packets in UDP ping", m_packetSize);
  cmd.AddValue ("meshInterfaces", "Number of radio interfaces used by each
meshHelper point. [1]", m_nIfaces);
  cmd.AddValue ("channels", "Use different frequency channels for different
meshInterfaces. [0]", m_chan);
  cmd.AddValue ("pcap", "Enable PCAP traces on meshInterfaces. [0]", m_pcap);
  cmd.AddValue ("stack", "Type of protocol stack. ns3::Dot11sStack by default",
m_stack);
  cmd.AddValue ("root", "Mac address of root meshHelper 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");

  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
}

void
```

```
MeshTest::CreateNodes ()
{

  // Node container creation (all node containers starts with "nc_")

  /*
   * Create m_ySize*m_xSize stations to form a grid topology
   */
  nc_all.Create (2);

  nc_mesh.Create (m_ySize * m_xSize);

  /// Needed to add mesh nodes and wireless nodes to the internet nodes
  nc_all.Add (nc_mesh);
  nc_mesh.Add (nc_all.Get (1)); /// n1---{meshHelper nodes}
  nc_wireless.Add (nc_all.Get (1));
  nc_wireless.Add (nc_mesh);
  nc_internet.Add (nc_all.Get (0));
  nc_internet.Add (nc_all.Get (1));
  nc_destination.Add (nc_all.Get (0));
  nc_destination.Add (nc_mesh);

  //Create P2P links between n0 <-> n1 (all devices starts with "de_")

  p2pHelper.SetDeviceAttribute ("DataRate", StringValue ("1Mbps"));
  p2pHelper.SetChannelAttribute ("Delay", StringValue ("10ms"));
  internetDevices = p2pHelper.Install (nc_internet);

  // Configure YansWifiChannel
  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
  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
   * meshHelper point device
   */
  meshHelper = MeshHelper::Default ();
  if (!Mac48Address (m_root.c_str ()).IsBroadcast ())
    {
      meshHelper.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
    meshHelper.SetStackInstaller (m_stack);
   }
  if (m_chan)
   {
    meshHelper.SetSpreadInterfaceChannels
(MeshHelper::SPREAD_CHANNELS);
   }
  else
   {
    meshHelper.SetSpreadInterfaceChannels (MeshHelper::ZERO_CHANNEL);
   }
  meshHelper.SetMacType ("RandomStart", TimeValue (Seconds
(m_randomStart)));
  // Set number of interfaces - default is single-interface meshHelper point
  meshHelper.SetNumberOfInterfaces (m_nIfaces);
  // Install protocols and return container if MeshPointDevices
  meshDevices = meshHelper.Install (wifiPhy, nc_mesh);
  // Setup mobility - static grid topology

#if 0
  MobilityHelper mobilityHelper;
  mobilityHelper.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"));
  mobilityHelper.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
"Bounds", RectangleValue(Rectangle(-1000, 1000, -1000, 1000)));
  mobilityHelper.Install (nc_mesh);
#endif

#if 0
  MobilityHelper mobility;
  Ptr<ListPositionAllocator> positionAlloc = CreateObject
<ListPositionAllocator>();


  positionAlloc ->Add(Vector(0, 0, 0)); // node1
  positionAlloc ->Add(Vector(37, 0, 0)); // node2
  positionAlloc ->Add(Vector(75, 75, 0)); // node3
  positionAlloc ->Add(Vector(150, 75, 0)); // node0
```

```
  positionAlloc ->Add(Vector(225, 75, 0)); // node1
  positionAlloc ->Add(Vector(75, 150, 0)); // node2
  positionAlloc ->Add(Vector(150, 150, 0)); // node3
  positionAlloc ->Add(Vector(225, 150, 0)); // node0
  positionAlloc ->Add(Vector(75, 0, 0)); // node1
  positionAlloc ->Add(Vector(150, 0, 0)); // node2
  positionAlloc ->Add(Vector(225, 0, 0)); // node3

  mobility.SetPositionAllocator(positionAlloc);
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(nc_all);

  Simulator::Schedule (Seconds (15.0), &SetPosition, nc_all.Get (3), 37.0, 75.0);


  AnimationInterface animation ("iMesh-handover.xml");
#endif



  if (m_pcap)
    wifiPhy.EnablePcapAll (std::string ("mp-"));

  /*AnimationInterface animation("iMesh.xml");

  animation.SetConstantPosition(nc_internet.Get(0), -200.0, 100.0);
  animation.SetConstantPosition(nc_internet.Get(1), -100.0, 100.0);*/
}

void
MeshTest::InstallInternetStack ()
{

  InternetStackHelper internetStackHelper;

  OlsrHelper routingProtocol;
  internetStackHelper.SetRoutingHelper (routingProtocol);

  internetStackHelper.Install (nc_mesh);
  internetStackHelper.Install (nc_all.Get (0)); /// Setting up protocol stack on node
0

  Ipv4AddressHelper meshAddress;
  meshAddress.SetBase ("10.1.1.0", "255.255.255.0");
  meshInterfaces = meshAddress.Assign (meshDevices);
```

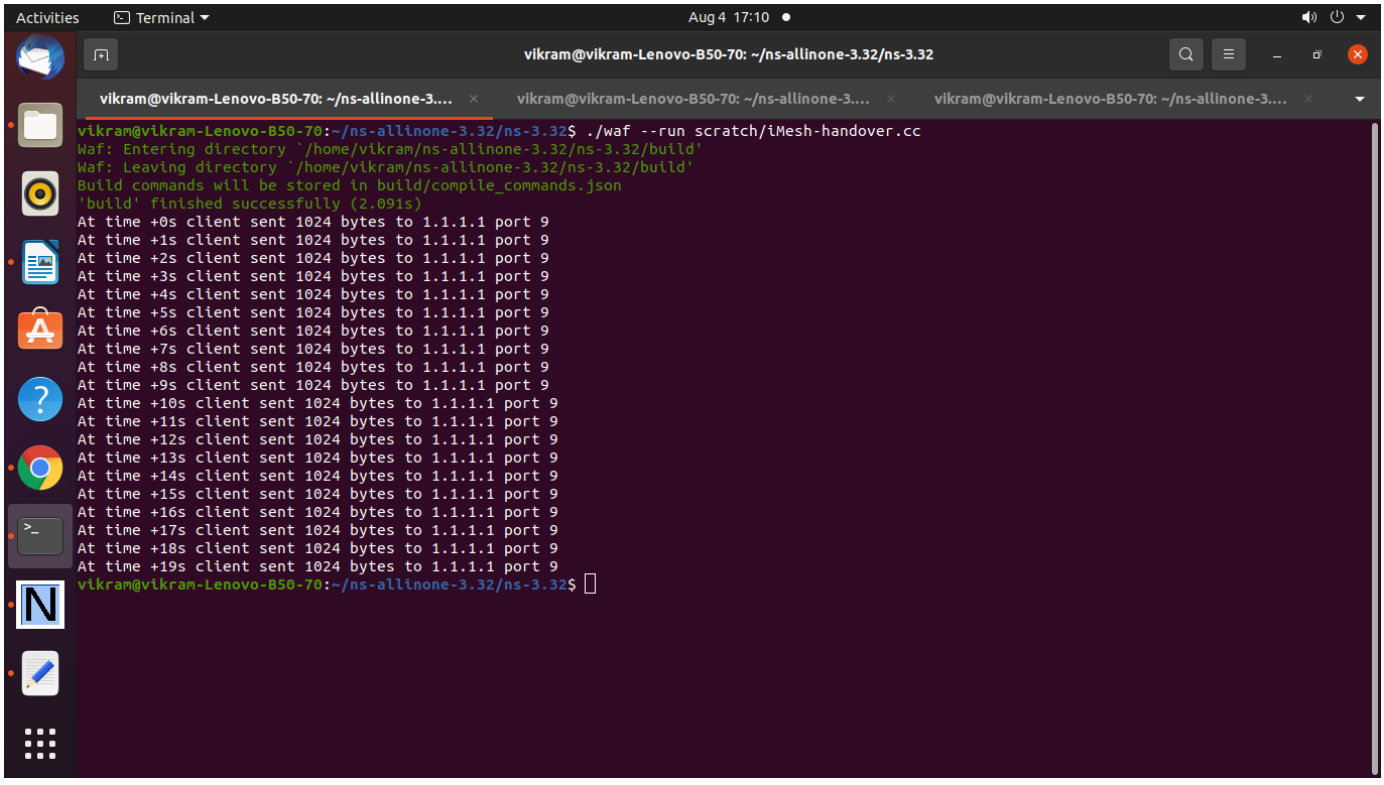
```
  Ipv4AddressHelper p2pAddress;
  p2pAddress.SetBase ("1.1.1.0", "255.255.255.0");
  p2pInterfaces = p2pAddress.Assign (internetDevices);

}

void
MeshTest::InstallApplication ()
{

  UdpEchoServerHelper echoServer (9);
  ApplicationContainer serverApps = echoServer.Install (nc_all.Get (0));
  serverApps.Start (Seconds (0.0));
  serverApps.Stop (Seconds (m_totalTime));

  UdpEchoClientHelper echoClient (p2pInterfaces.GetAddress (0), 9);
  //echoClient.SetAttribute ("MaxPackets", UintegerValue ((uint32_t)
(m_totalTime * (1 / m_packetInterval))));
  echoClient.SetAttribute ("MaxPackets", UintegerValue (20));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (m_packetInterval)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (m_packetSize));

  ApplicationContainer clientApps1 = echoClient.Install (nc_mesh.Get (m_xSize *
m_ySize - 1));
  //ApplicationContainer clientApps2 = echoClient.Install (nc_mesh.Get (m_xSize
* m_ySize - 2));



  clientApps1.Start (Seconds (0.0));
  clientApps1.Stop (Seconds (m_totalTime));

  //  clientApps2.Start (Seconds (12));
  //  clientApps2.Stop (Seconds (m_totalTime));

  /*for (tmp_x = 2; tmp_x < m_xSize * m_ySize + 2; tmp_x++)
  {
    sprintf (tmp_char, "%d-STA", tmp_x);
    AnimationInterface::SetNodeDescription (nc_all.Get (tmp_x), tmp_char);
  }
  AnimationInterface::SetNodeDescription (nc_all.Get (0), "0-Internet");
  AnimationInterface::SetNodeDescription (nc_all.Get (1), "1-AP");
  AnimationInterface::SetNodeColor (nc_mesh, 0, 255, 0);
  AnimationInterface::SetNodeColor (nc_internet.Get (0), 255, 0, 0);
```

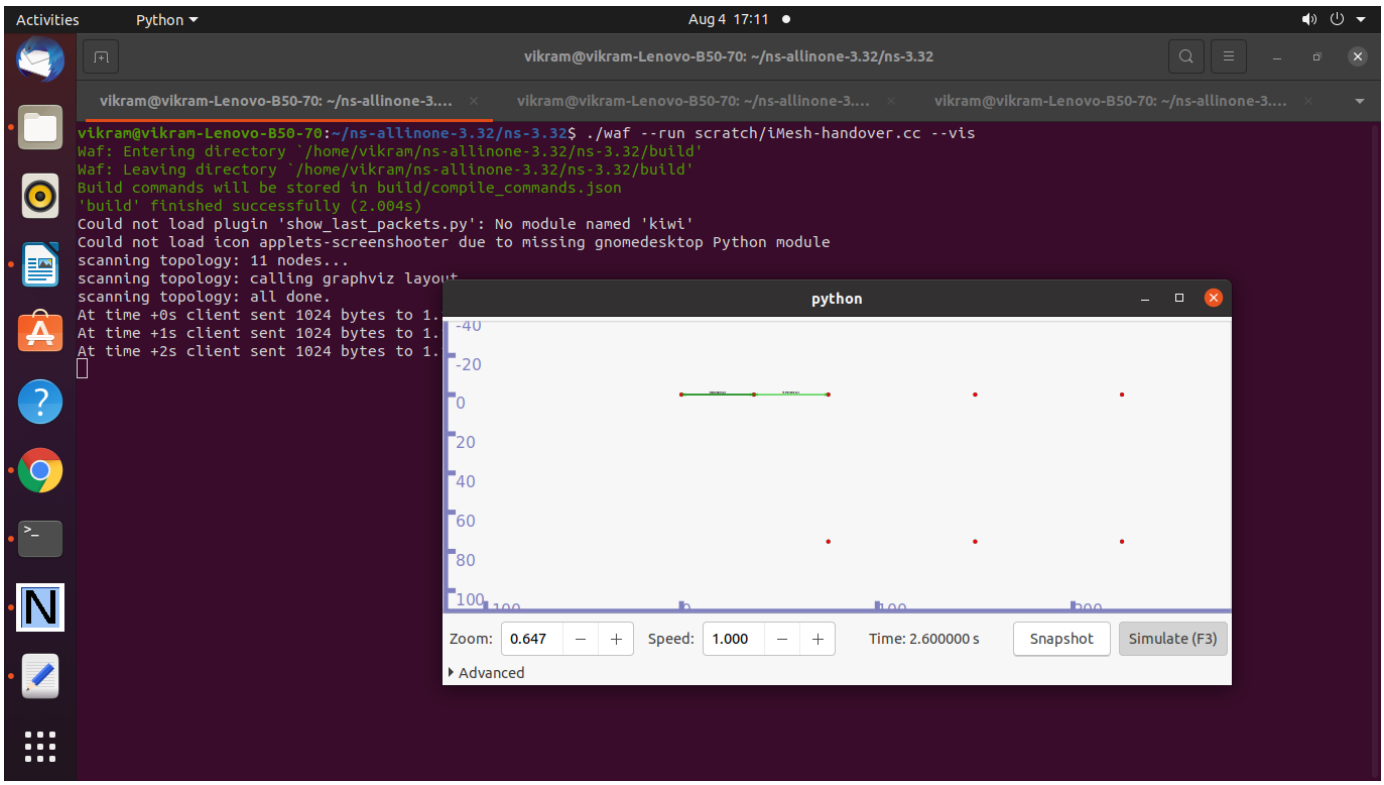
```
  AnimationInterface::SetNodeColor (nc_internet.Get (1), 0, 0, 255);

   */



}

int
MeshTest::Run ()
{
  CreateNodes ();
  InstallInternetStack ();
  InstallApplication ();


  Ptr<ListPositionAllocator> positionAlloc = CreateObject
<ListPositionAllocator>();


  positionAlloc ->Add(Vector(0, 0, 0)); // node1
  positionAlloc ->Add(Vector(37, 0, 0)); // node2
  positionAlloc ->Add(Vector(75, 75, 0)); // node3
  positionAlloc ->Add(Vector(150, 75, 0)); // node0
  positionAlloc ->Add(Vector(225, 75, 0)); // node1
  positionAlloc ->Add(Vector(75, 150, 0)); // node2
  positionAlloc ->Add(Vector(150, 150, 0)); // node3
  positionAlloc ->Add(Vector(225, 150, 0)); // node0
  positionAlloc ->Add(Vector(75, 0, 0)); // node1
  positionAlloc ->Add(Vector(150, 0, 0)); // node2
  positionAlloc ->Add(Vector(225, 0, 0)); // node3

  mobility.SetPositionAllocator(positionAlloc);
  mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
  mobility.Install(nc_all);

  Simulator::Schedule (Seconds (12.0), &SetPosition, nc_mesh.Get (m_xSize *
m_ySize - 1), 37.0, 75.0);

  Simulator::Stop (Seconds (m_totalTime));

  AnimationInterface animation ("iMesh-handover.xml");

  animation.EnablePacketMetadata (false);
```

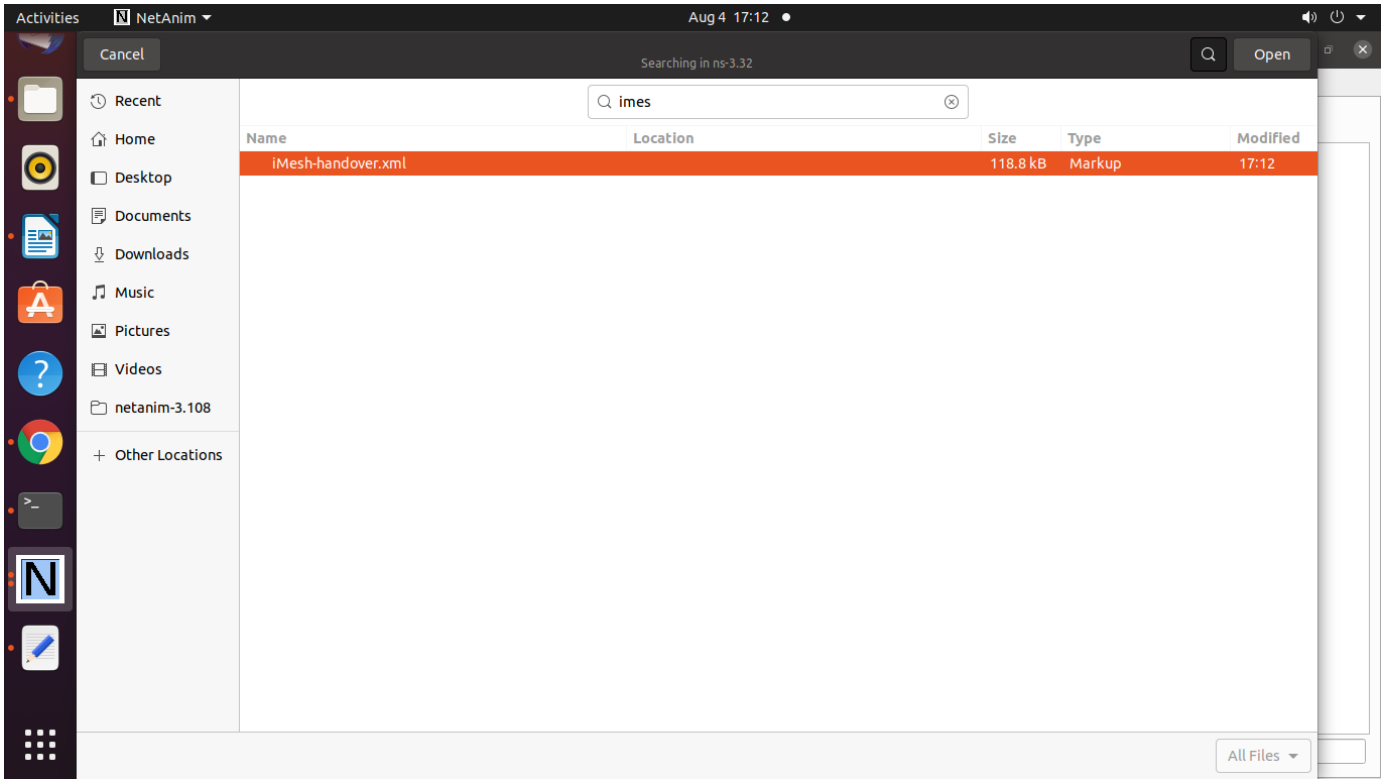
```
  animation.EnableIpv4RouteTracking ("mesh-handover-route.xml", Seconds (0),
Seconds (50), Seconds (100));


//  animation.SetConstantPosition (nc_all.Get (0), 0.0, 0.0);
//  animation.SetConstantPosition (nc_internet.Get (1), 37.0, 0.0);
//  animation.SetConstantPosition (nc_mesh.Get (0), 75.0, 75.0);
//  animation.SetConstantPosition (nc_mesh.Get (1), 150.0, 75.0);
//  animation.SetConstantPosition (nc_mesh.Get (2), 225.0, 75.0);
//  animation.SetConstantPosition (nc_mesh.Get (3), 75.0, 150.0);
//  animation.SetConstantPosition (nc_mesh.Get (4), 150.0, 150.0);
//  animation.SetConstantPosition (nc_mesh.Get (5), 225.0, 150.0);
//  animation.SetConstantPosition (nc_mesh.Get (6), 75.0, 0.0);
//  animation.SetConstantPosition (nc_mesh.Get (7), 150.0, 0.0);
//  animation.SetConstantPosition (nc_mesh.Get (8), 225.0, 0.0);

  //animation.EnablePacketMetadata(false);
  //animation.EnableIpv4RouteTracking ("iMesh-handover.xml", Seconds (0),
Seconds (m_totalTime), Seconds (0.25));




  //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-handover-true-" << n << ".xml";
       std::cerr << "Printing meshHelper point device #" << n << " diagnostics to "
<< os.str () << "\n";
       std::ofstream of;
       of.open (os.str ().c_str ());
       if (!of.is_open ())
```

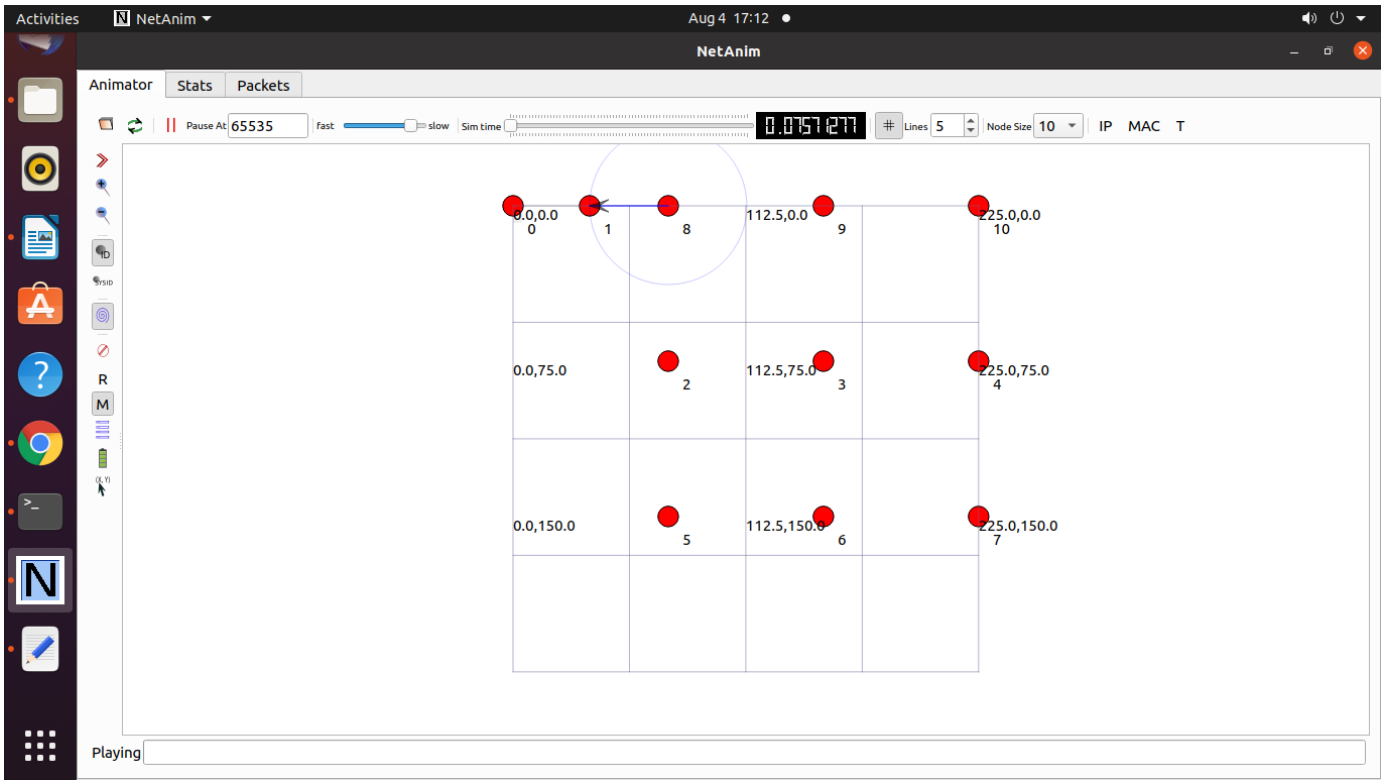
```
      {
        std::cerr << "Error: Can't open file " << os.str () << "\n";
        return;
      }
    meshHelper.Report (*i, of);
    of.close ();
  }
}

int
main (int argc, char *argv[])
{
  ns3::PacketMetadata::Enable ();
  MeshTest t;
  t.Configure (argc, argv);
  return t.Run ();
}
```

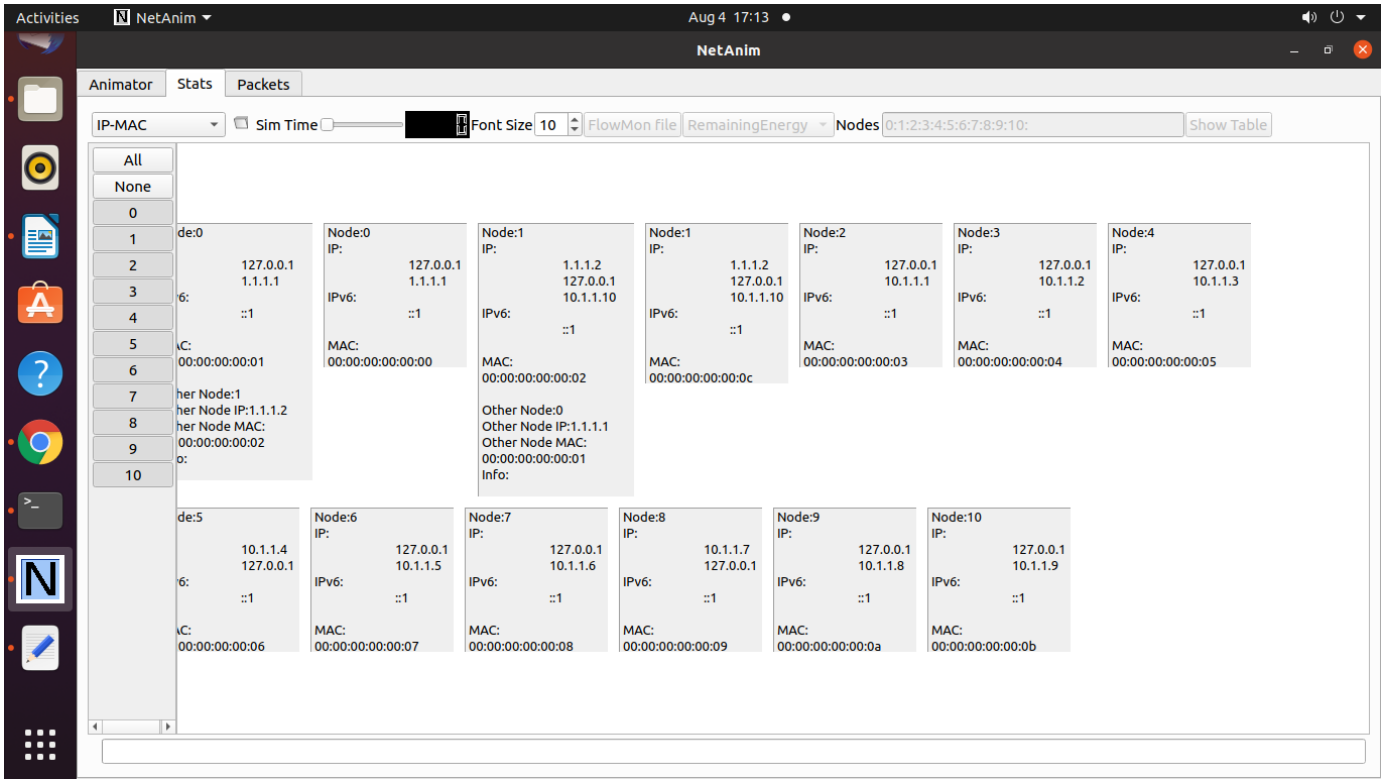
Output :

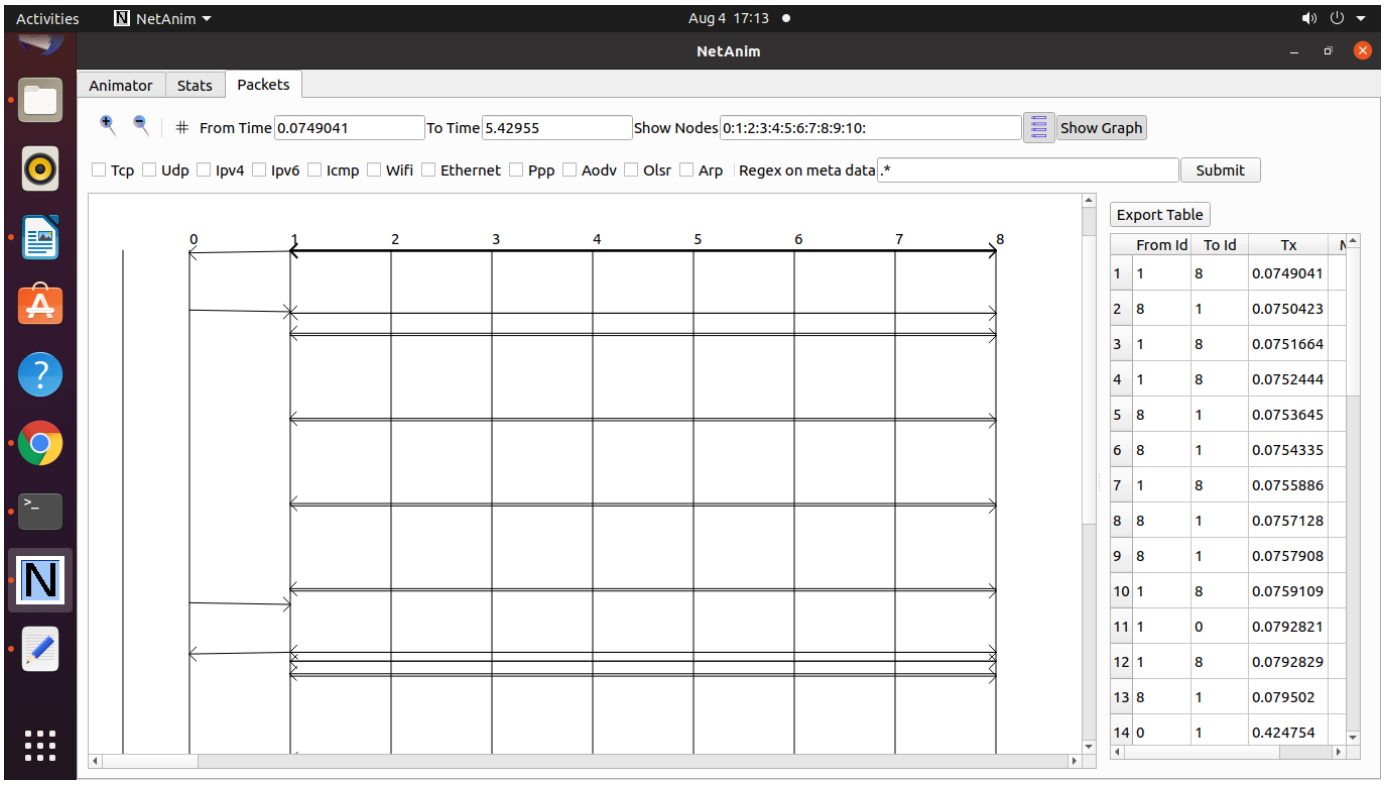



Activities
Python
Aug 4 17:11
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3....
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/iMesh-handover.cc --vis
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.004s)
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: 11 nodes...
scanning topology: calling graphviz layout
scanning topology: all done.
At time +0s client sent 1024 bytes to 1.
At time +1s client sent 1024 bytes to 1.
At time +2s client sent 1024 bytes to 1.
python
-40
-20
0
20
40
60
80
100
Zoom: 0.647
Speed: 1.000
Time: 2.600000 s
Snapshot
Simulate (F3)
Advanced


Activities
NetAnim
Aug 4 17:12
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
imes
Name
Location
Size
Type
Modified
iMesh-handover.xml
118.8 kB
Markup
17:12
All Files


Activities
NetAnim
Aug 4 17:12
NetAnim
Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
Lines 5
Node Size 10
IP
MAC
T
0.0,0.0
0
1
8
112.5,0.0
9
225.0,0.0
10
0.0,75.0
2
112.5,75.0
3
225.0,75.0
4
0.0,150.0
5
112.5,150.0
6
225.0,150.0
7
Playing


Activities
NetAnim
Aug 4 17:13
NetAnim
Animator
Stats
Packets
IP-MAC
Sim Time
Font Size 10
FlowMon file
RemainingEnergy
Nodes 0:1:2:3:4:5:6:7:8:9:10:
Show Table
All
None
0
1
2
3
4
5
6
7
8
9
10
Node:0
IP:
127.0.0.1
1.1.1.1
IPv6:
::1
MAC:
00:00:00:00:00:00
Node:1
IP:
1.1.1.2
127.0.0.1
10.1.1.10
IPv6:
::1
MAC:
00:00:00:00:00:02
Other Node:0
Other Node IP:1.1.1.1
Other Node MAC:
00:00:00:00:00:01
Info:
Node:1
IP:
1.1.1.2
127.0.0.1
10.1.1.10
IPv6:
::1
MAC:
00:00:00:00:00:0c
Node:2
IP:
127.0.0.1
10.1.1.1
IPv6:
::1
MAC:
00:00:00:00:00:03
Node:3
IP:
127.0.0.1
10.1.1.2
IPv6:
::1
MAC:
00:00:00:00:00:04
Node:4
IP:
127.0.0.1
10.1.1.3
IPv6:
::1
MAC:
00:00:00:00:00:05
Node:6
IP:
127.0.0.1
10.1.1.5
IPv6:
::1
MAC:
00:00:00:00:00:07
Node:7
IP:
127.0.0.1
10.1.1.6
IPv6:
::1
MAC:
00:00:00:00:00:08
Node:8
IP:
10.1.1.7
127.0.0.1
IPv6:
::1
MAC:
00:00:00:00:00:09
Node:9
IP:
127.0.0.1
10.1.1.8
IPv6:
::1
MAC:
00:00:00:00:00:0a
Node:10
IP:
127.0.0.1
10.1.1.9
IPv6:
::1
MAC:
00:00:00:00:00:0b



	From Id	To Id	Tx
1	1	8	0.0749041
2	8	1	0.0750423
3	1	8	0.0751664
4	1	8	0.0752444
5	8	1	0.0753645
6	8	1	0.0754335
7	1	8	0.0755886
8	8	1	0.0757128
9	8	1	0.0757908
10	1	8	0.0759109
11	1	0	0.0792821
12	1	8	0.0792829
13	8	1	0.079502
14	0	1	0.424754

Program 22 : Program to simulate FTP using TCP protocol.

tcp-large-transfer.cc

```
/* -*- 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
//
//         10Mb/s, 10ms       10Mb/s, 10ms
//       n0-----------------n1-----------------n2
//
//
// - Tracing of queues and packet receptions to file
//   "tcp-large-transfer.tr"
// - pcap traces also generated in the following files
//   "tcp-large-transfer-$n-$i.pcap" where n and i represent node and interface
// numbers respectively
//  Usage (e.g.): ./waf --run tcp-large-transfer

#include <iostream>
#include <fstream>
#include <string>

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

```
#include "ns3/point-to-point-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("TcpLargeTransfer");

// The number of bytes to send in this simulation.
static const uint32_t totalTxBytes = 2000000;
static uint32_t currentTxBytes = 0;
// Perform series of 1040 byte writes (this is a multiple of 26 since
// we want to detect data splicing in the output stream)
static const uint32_t writeSize = 1040;
uint8_t data[writeSize];
std::string animFile = "ftp-animation.xml";

// These are for starting the writing process, and handling the sending
// socket's notification upcalls (events).  These two together more or less
// implement a sending "Application", although not a proper ns3::Application
// subclass.

void StartFlow (Ptr<Socket>, Ipv4Address, uint16_t);
void WriteUntilBufferFull (Ptr<Socket>, uint32_t);

static void
CwndTracer (uint32_t oldval, uint32_t newval)
{
  NS_LOG_INFO ("Moving cwnd from " << oldval << " to " << newval);
}

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("TcpL4Protocol", LOG_LEVEL_ALL);
  //  LogComponentEnable("TcpSocketImpl", LOG_LEVEL_ALL);
  //  LogComponentEnable("PacketSink", LOG_LEVEL_ALL);
  //  LogComponentEnable("TcpLargeTransfer", LOG_LEVEL_ALL);

  CommandLine cmd (__FILE__);
  cmd.Parse (argc, argv);

  // initialize the tx buffer.
  for(uint32_t i = 0; i < writeSize; ++i)
```

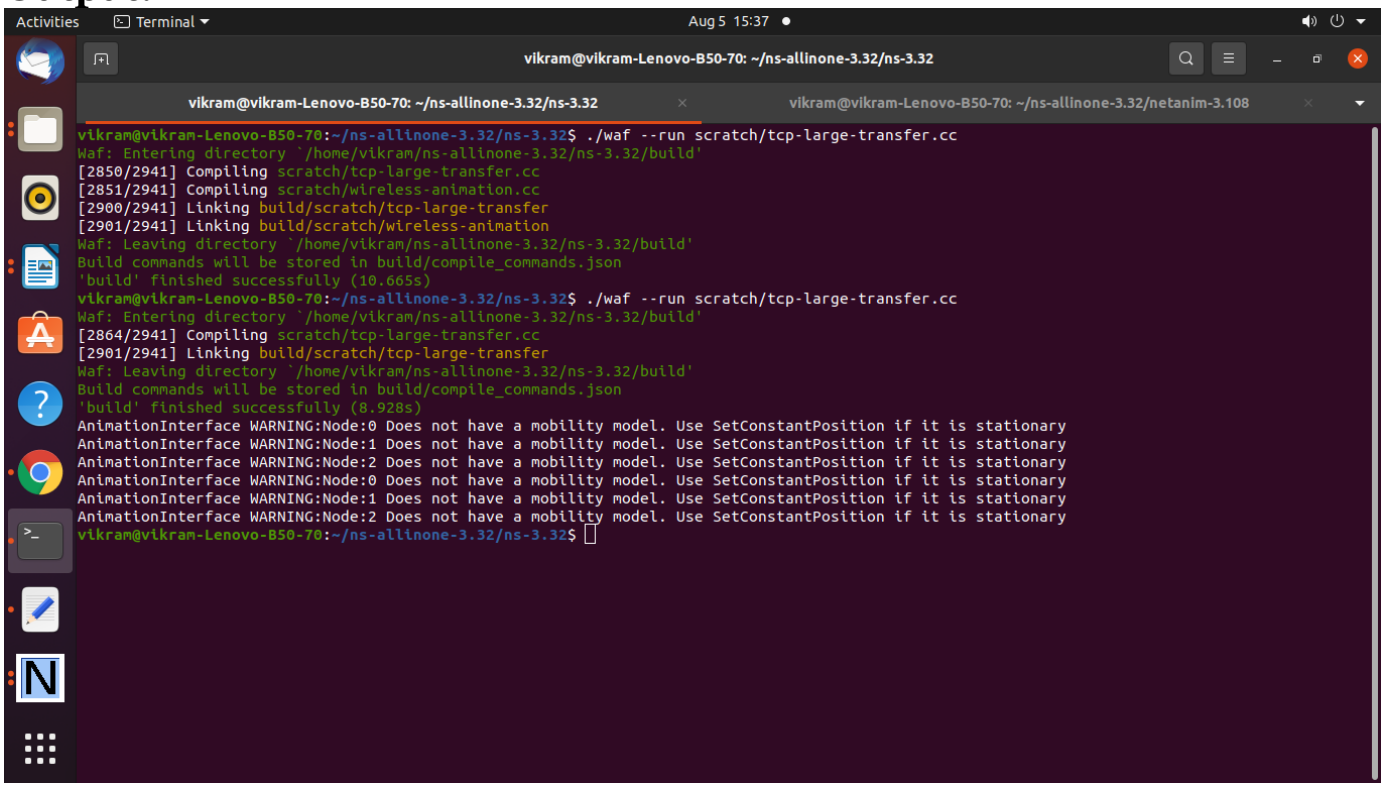
```
 {
  char m = toascii (97 + i % 26);
  data[i] = m;
 }

// Here, we will explicitly create three nodes.  The first container contains
// nodes 0 and 1 from the diagram above, and the second one contains nodes
// 1 and 2.  This reflects the channel connectivity, and will be used to
// install the network interfaces and connect them with a channel.
NodeContainer n0n1;
n0n1.Create (2);

NodeContainer n1n2;
n1n2.Add (n0n1.Get (1));
n1n2.Create (1);

// We create the channels first without any IP addressing information
// First make and configure the helper, so that it will put the appropriate
// attributes on the network interfaces and channels we are about to install.
PointToPointHelper p2p;
p2p.SetDeviceAttribute ("DataRate", DataRateValue (DataRate (10000000)));
p2p.SetChannelAttribute ("Delay", TimeValue (MilliSeconds (10)));

// And then install devices and channels connecting our topology.
NetDeviceContainer dev0 = p2p.Install (n0n1);
NetDeviceContainer dev1 = p2p.Install (n1n2);

// Now add ip/tcp stack to all nodes.
InternetStackHelper internet;
internet.InstallAll ();

// Later, we add IP addresses.
Ipv4AddressHelper ipv4;
ipv4.SetBase ("10.1.3.0", "255.255.255.0");
ipv4.Assign (dev0);
ipv4.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer ipInterfs = ipv4.Assign (dev1);

// and setup ip routing tables to get total ip-level connectivity.
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

////////////////////////////////////////////////////////////////////
// Simulation 1
//
// Send 2000000 bytes over a connection to server port 50000 at time 0
```

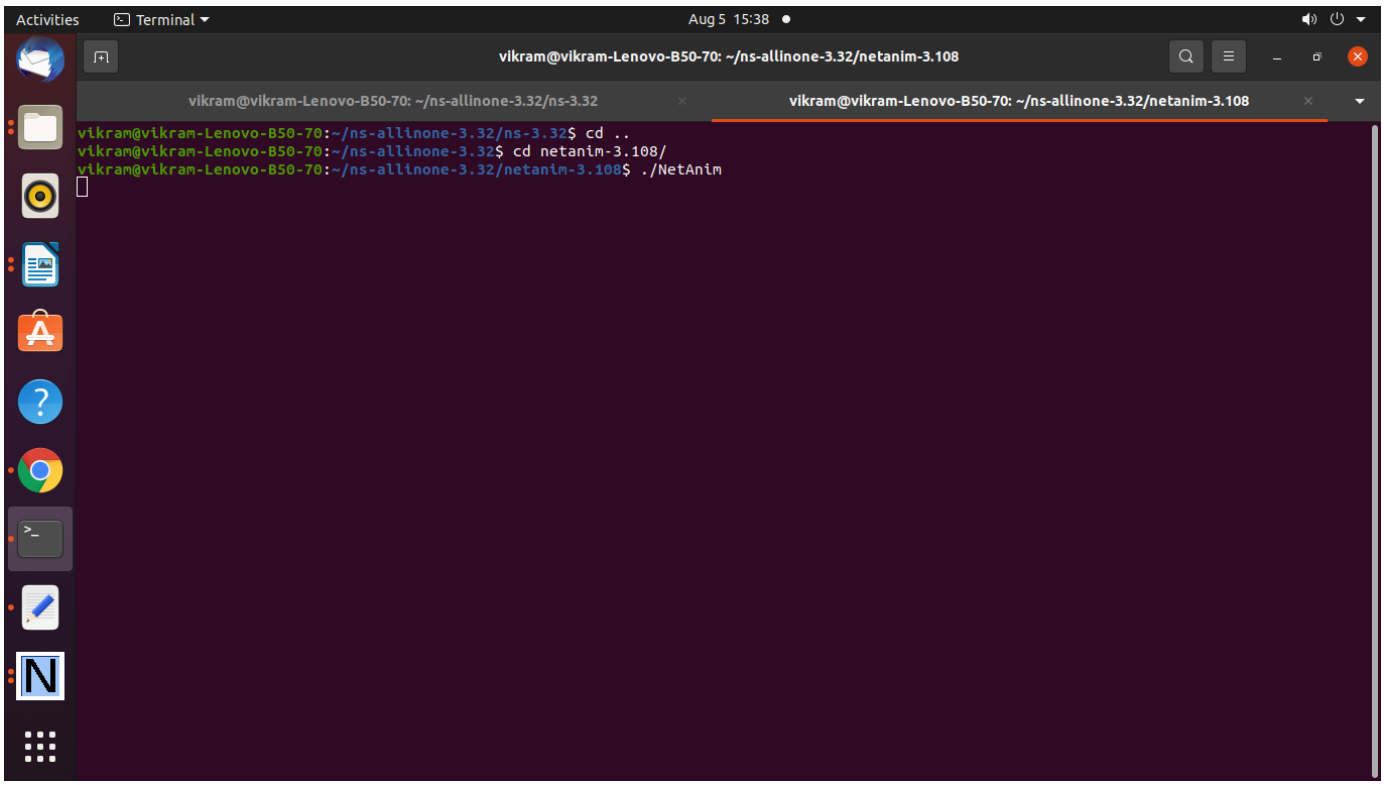
```
  // Should observe SYN exchange, a lot of data segments and ACKS, and FIN
  // exchange.  FIN exchange isn't quite compliant with TCP spec (see release
  // notes for more info)
  //
  ////////////////////////////////////////////////////////////////////////////

  uint16_t servPort = 50000;

  // Create a packet sink to receive these packets on n2...
  PacketSinkHelper sink ("ns3::TcpSocketFactory",
                         InetSocketAddress (Ipv4Address::GetAny (), servPort));

  ApplicationContainer apps = sink.Install (n1n2.Get (1));
  apps.Start (Seconds (0.0));
  apps.Stop (Seconds (3.0));

  // Create a source to send packets from n0.  Instead of a full Application
  // and the helper APIs you might see in other example files, this example
  // will use sockets directly and register some socket callbacks as a sending
  // "Application".

  // Create and bind the socket...
  Ptr<Socket> localSocket =
    Socket::CreateSocket (n0n1.Get (0), TcpSocketFactory::GetTypeId ());
  localSocket->Bind ();

  // Trace changes to the congestion window
  Config::ConnectWithoutContext
("/NodeList/0/$ns3::TcpL4Protocol/SocketList/0/CongestionWindow",
MakeCallback (&CwndTracer));

  // ...and schedule the sending "Application"; This is similar to what an
  // ns3::Application subclass would do internally.
  Simulator::ScheduleNow (&StartFlow, localSocket,
                          ipInterfs.GetAddress (1), servPort);

  // One can toggle the comment for the following line on or off to see the
  // effects of finite send buffer modelling.  One can also change the size of
  // said buffer.

  //localSocket->SetAttribute("SndBufSize", UintegerValue(4096));

  //Ask for ASCII and pcap traces of network traffic
  AsciiTraceHelper ascii;
  p2p.EnableAsciiAll (ascii.CreateFileStream ("tcp-large-transfer.tr"));
```

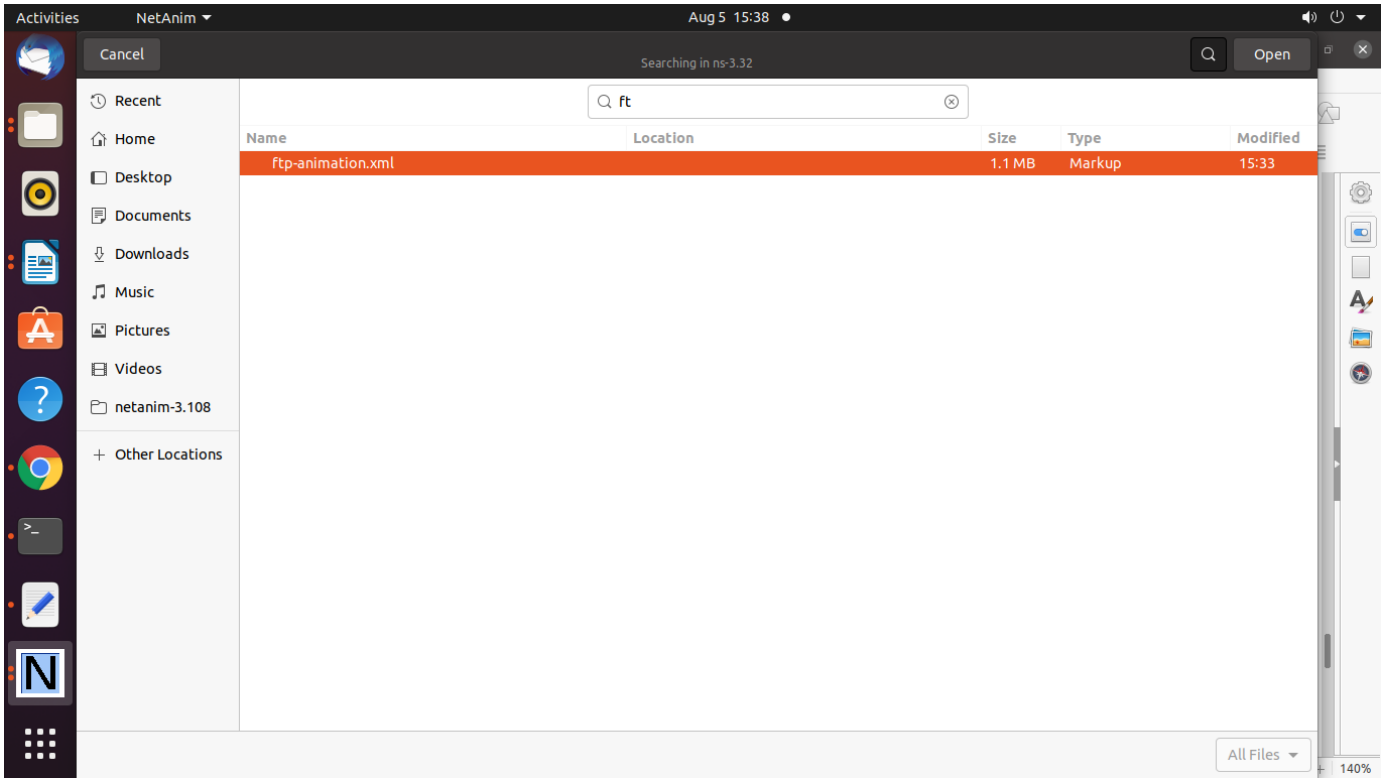
```
  p2p.EnablePcapAll ("tcp-large-transfer");
   // Create the animation object and configure for specified output
     AnimationInterface anim (animFile);
  // Finally, set up the simulator to run.  The 1000 second hard limit is a
  // failsafe in case some change above causes the simulation to never end
  Simulator::Stop (Seconds (1000));
  Simulator::Run ();
  Simulator::Destroy ();
}


//----------------------------------------------------------------------
//----------------------------------------------------------------------
//----------------------------------------------------------------------
//begin implementation of sending "Application"
void StartFlow (Ptr<Socket> localSocket,
                Ipv4Address servAddress,
                uint16_t servPort)
{
  NS_LOG_LOGIC ("Starting flow at time " <<  Simulator::Now ().GetSeconds ());
  localSocket->Connect (InetSocketAddress (servAddress, servPort)); //connect

  // tell the tcp implementation to call WriteUntilBufferFull again
  // if we blocked and new tx buffer space becomes available
  localSocket->SetSendCallback (MakeCallback (&WriteUntilBufferFull));
  WriteUntilBufferFull (localSocket, localSocket->GetTxAvailable ());
}

void WriteUntilBufferFull (Ptr<Socket> localSocket, uint32_t txSpace)
{
  while (currentTxBytes < totalTxBytes && localSocket->GetTxAvailable () > 0)
    {
      uint32_t left = totalTxBytes - currentTxBytes;
      uint32_t dataOffset = currentTxBytes % writeSize;
      uint32_t toWrite = writeSize - dataOffset;
      toWrite = std::min (toWrite, left);
      toWrite = std::min (toWrite, localSocket->GetTxAvailable ());
      int amountSent = localSocket->Send (&data[dataOffset], toWrite, 0);
      if(amountSent < 0)
        {
          // we will be called again when new tx space becomes available.
          return;
        }
      currentTxBytes += amountSent;
    }
```

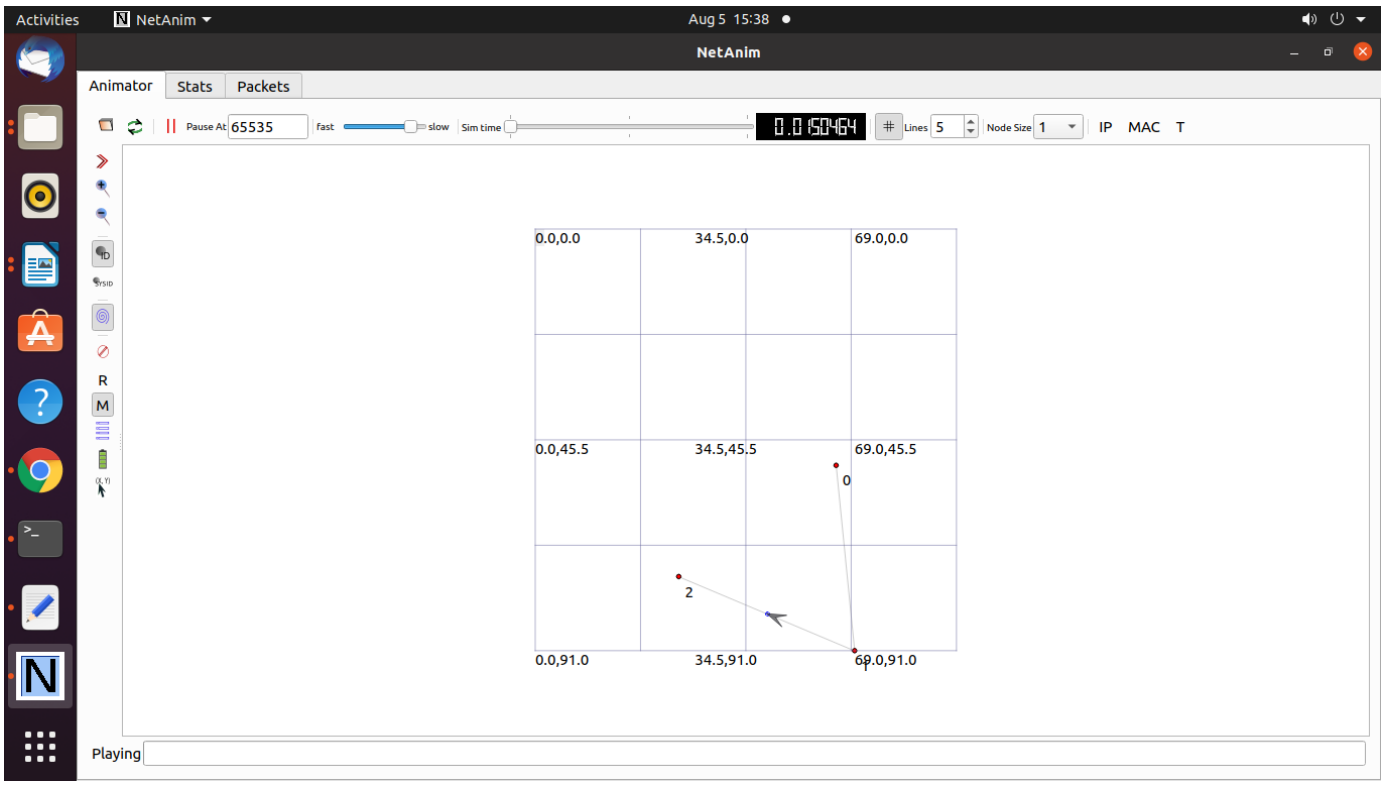
```
  if (currentTxBytes >= totalTxBytes)
    {
      localSocket->Close ();
    }
}
```

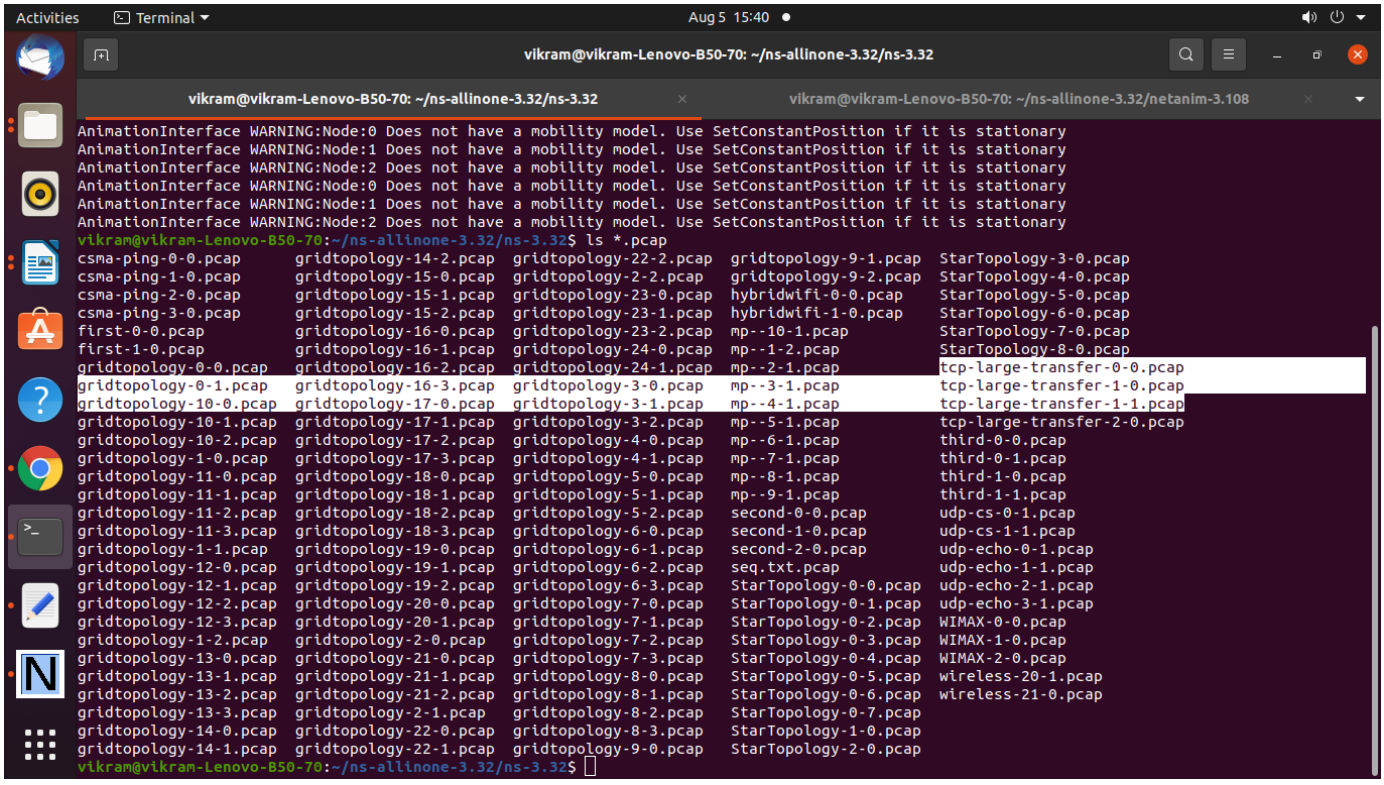
Output:

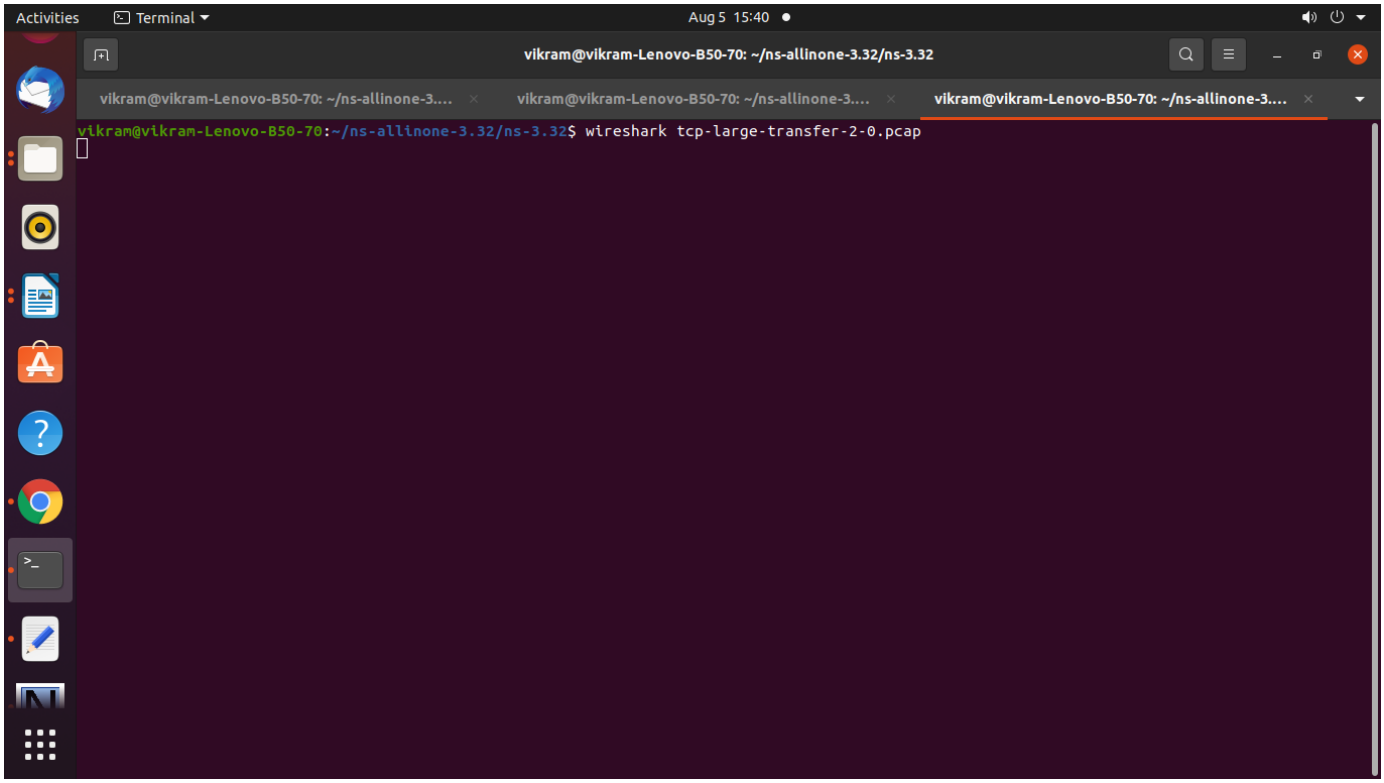


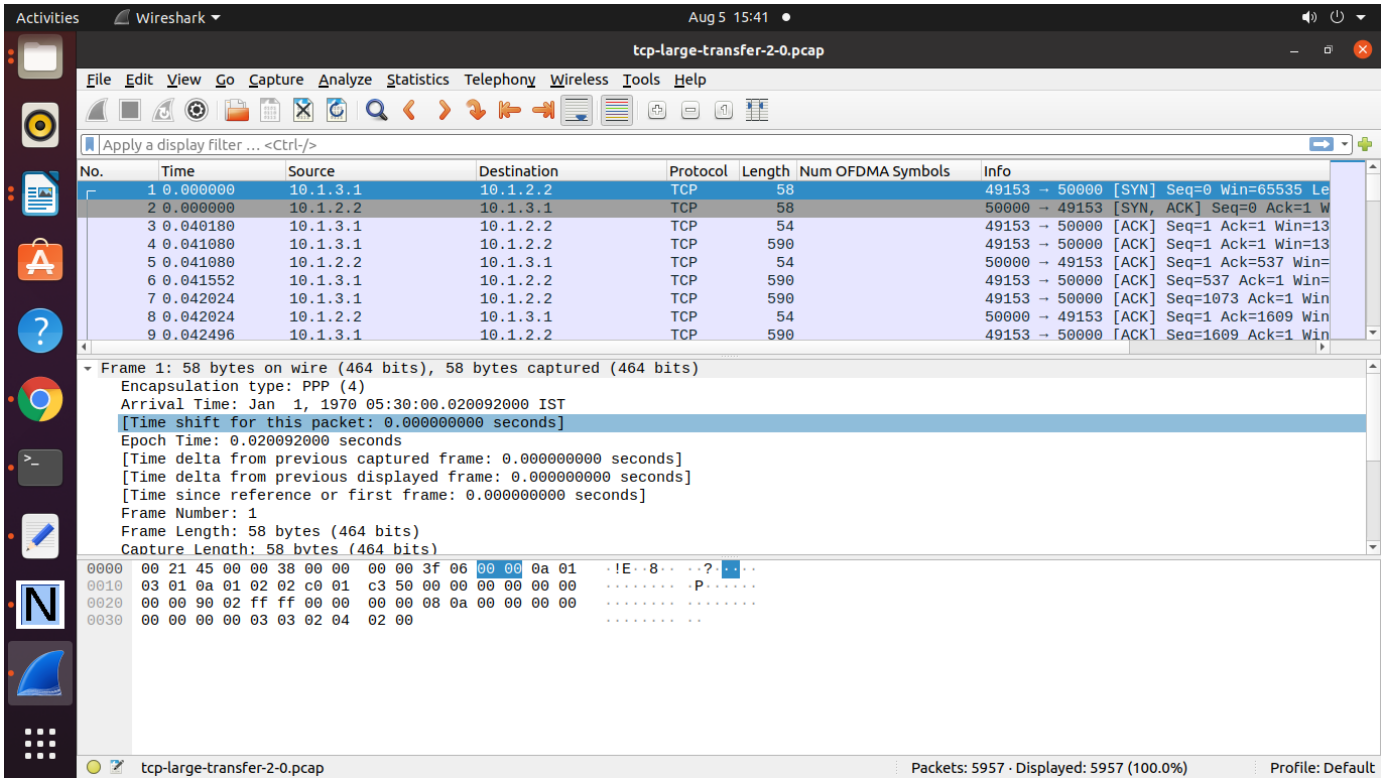

Activities
Terminal
Aug 5 15:38
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/netanim-3.108
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/netanim-3.108
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ cd ..
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32$ cd netanim-3.108/
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/netanim-3.108$ ./NetAnim


Activities
NetAnim
Aug 5 15:38
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
ft
Name
Location
Size
Type
Modified
ftp-animation.xml
1.1 MB
Markup
15:33
All Files
140%


Activities
NetAnim
Aug 5 15:38
NetAnim
Animator
Stats
Packets
Pause At
65535
fast
slow
Sim time
Lines
5
Node Size
1
IP
MAC
T
0.0,0.0
34.5,0.0
69.0,0.0
0.0,45.5
34.5,45.5
69.0,45.5
0.0,91.0
34.5,91.0
69.0,91.0
0
2
Playing

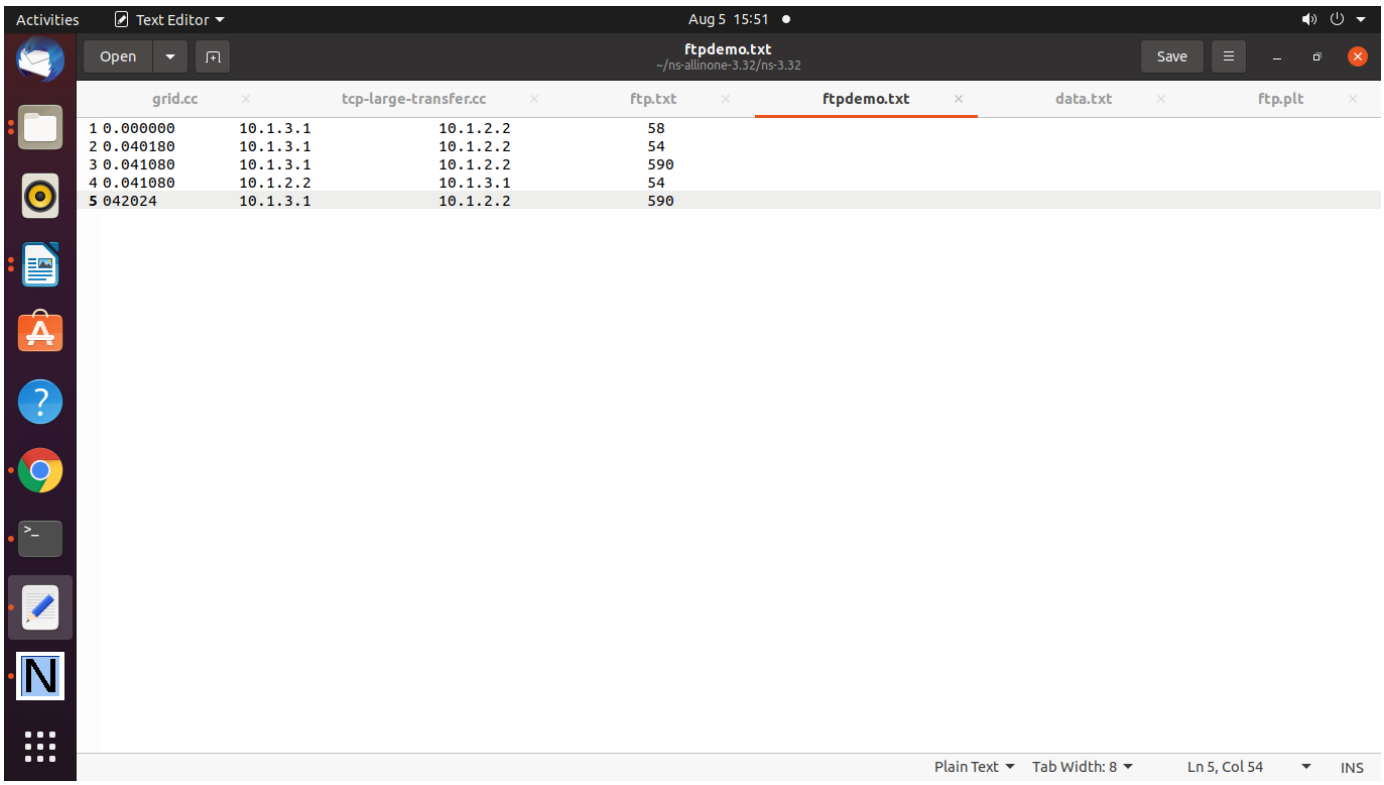

Activities
Terminal
Aug 5 15:40
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/netanim-3.108
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ls *.pcap
csma-ping-0-0.pcap gridtopology-14-2.pcap gridtopology-22-2.pcap gridtopology-9-1.pcap StarTopology-3-0.pcap
csma-ping-1-0.pcap gridtopology-15-0.pcap gridtopology-2-2.pcap gridtopology-9-2.pcap StarTopology-4-0.pcap
csma-ping-2-0.pcap gridtopology-15-1.pcap gridtopology-23-0.pcap hybridwifi-0-0.pcap StarTopology-5-0.pcap
csma-ping-3-0.pcap gridtopology-15-2.pcap gridtopology-23-1.pcap hybridwifi-1-0.pcap StarTopology-6-0.pcap
first-0-0.pcap gridtopology-16-0.pcap gridtopology-23-2.pcap mp--10-1.pcap StarTopology-7-0.pcap
first-1-0.pcap gridtopology-16-1.pcap gridtopology-24-0.pcap mp--1-2.pcap StarTopology-8-0.pcap
gridtopology-0-0.pcap gridtopology-16-2.pcap gridtopology-24-1.pcap mp--2-1.pcap tcp-large-transfer-0-0.pcap
gridtopology-0-1.pcap gridtopology-16-3.pcap gridtopology-3-0.pcap mp--3-1.pcap tcp-large-transfer-1-0.pcap
gridtopology-10-0.pcap gridtopology-17-0.pcap gridtopology-3-1.pcap mp--4-1.pcap tcp-large-transfer-1-1.pcap
gridtopology-10-1.pcap gridtopology-17-1.pcap gridtopology-3-2.pcap mp--5-1.pcap tcp-large-transfer-2-0.pcap
gridtopology-10-2.pcap gridtopology-17-2.pcap gridtopology-4-0.pcap mp--6-1.pcap third-0-0.pcap
gridtopology-1-0.pcap gridtopology-17-3.pcap gridtopology-4-1.pcap mp--7-1.pcap third-0-1.pcap
gridtopology-11-0.pcap gridtopology-18-0.pcap gridtopology-5-0.pcap mp--8-1.pcap third-1-0.pcap
gridtopology-11-1.pcap gridtopology-18-1.pcap gridtopology-5-1.pcap mp--9-1.pcap third-1-1.pcap
gridtopology-11-2.pcap gridtopology-18-2.pcap gridtopology-5-2.pcap second-0-0.pcap udp-cs-0-1.pcap
gridtopology-11-3.pcap gridtopology-18-3.pcap gridtopology-6-0.pcap second-1-0.pcap udp-cs-1-1.pcap
gridtopology-1-1.pcap gridtopology-19-0.pcap gridtopology-6-1.pcap second-2-0.pcap udp-echo-0-1.pcap
gridtopology-12-0.pcap gridtopology-19-1.pcap gridtopology-6-2.pcap seq.txt.pcap udp-echo-1-1.pcap
gridtopology-12-1.pcap gridtopology-19-2.pcap gridtopology-6-3.pcap StarTopology-0-0.pcap udp-echo-2-1.pcap
gridtopology-12-2.pcap gridtopology-20-0.pcap gridtopology-7-0.pcap StarTopology-0-1.pcap udp-echo-3-1.pcap
gridtopology-12-3.pcap gridtopology-20-1.pcap gridtopology-7-1.pcap StarTopology-0-2.pcap WIMAX-0-0.pcap
gridtopology-1-2.pcap gridtopology-2-0.pcap gridtopology-7-2.pcap StarTopology-0-3.pcap WIMAX-1-0.pcap
gridtopology-13-0.pcap gridtopology-21-0.pcap gridtopology-7-3.pcap StarTopology-0-4.pcap WIMAX-2-0.pcap
gridtopology-13-1.pcap gridtopology-21-1.pcap gridtopology-8-0.pcap StarTopology-0-5.pcap wireless-20-1.pcap
gridtopology-13-2.pcap gridtopology-21-2.pcap gridtopology-8-1.pcap StarTopology-0-6.pcap wireless-21-0.pcap
gridtopology-13-3.pcap gridtopology-2-1.pcap gridtopology-8-2.pcap StarTopology-0-7.pcap
gridtopology-14-0.pcap gridtopology-22-0.pcap gridtopology-8-3.pcap StarTopology-1-0.pcap
gridtopology-14-1.pcap gridtopology-22-1.pcap gridtopology-9-0.pcap StarTopology-2-0.pcap
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$



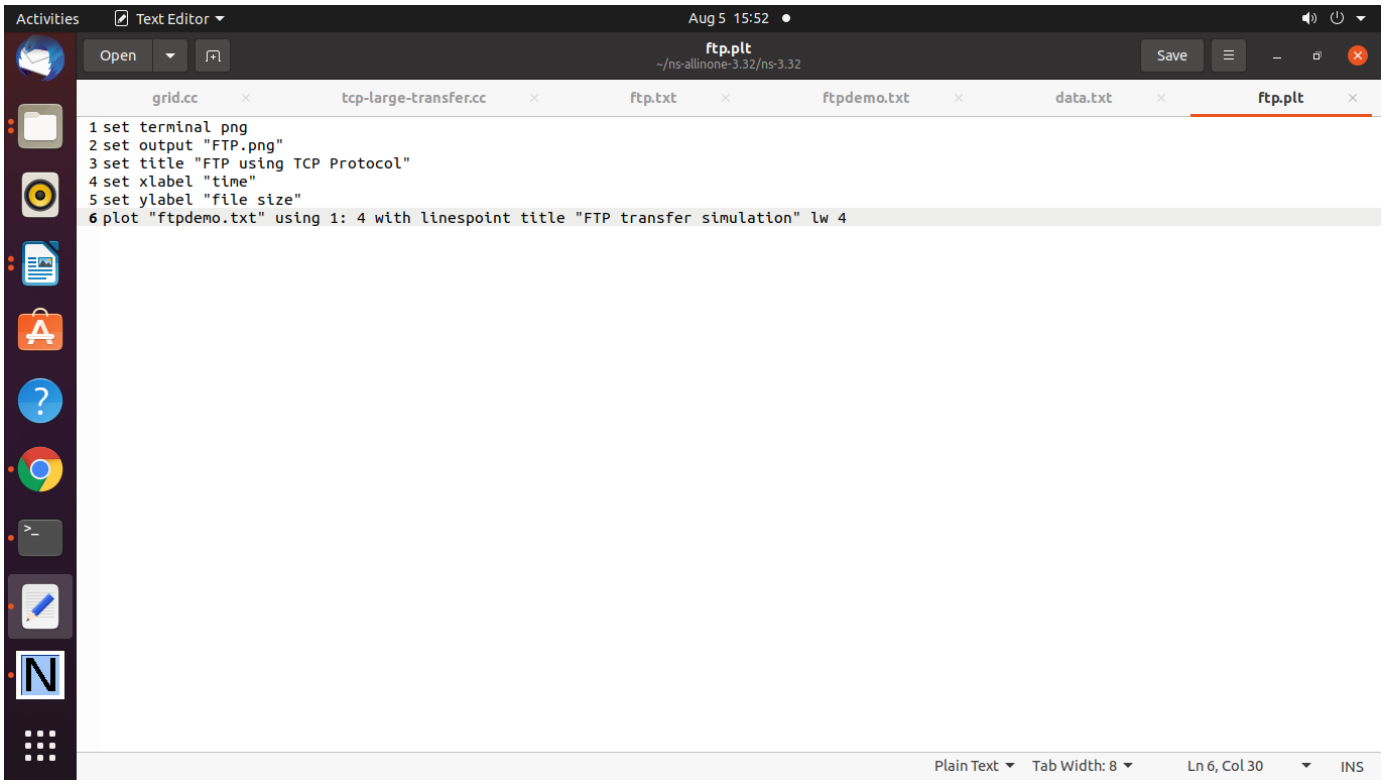


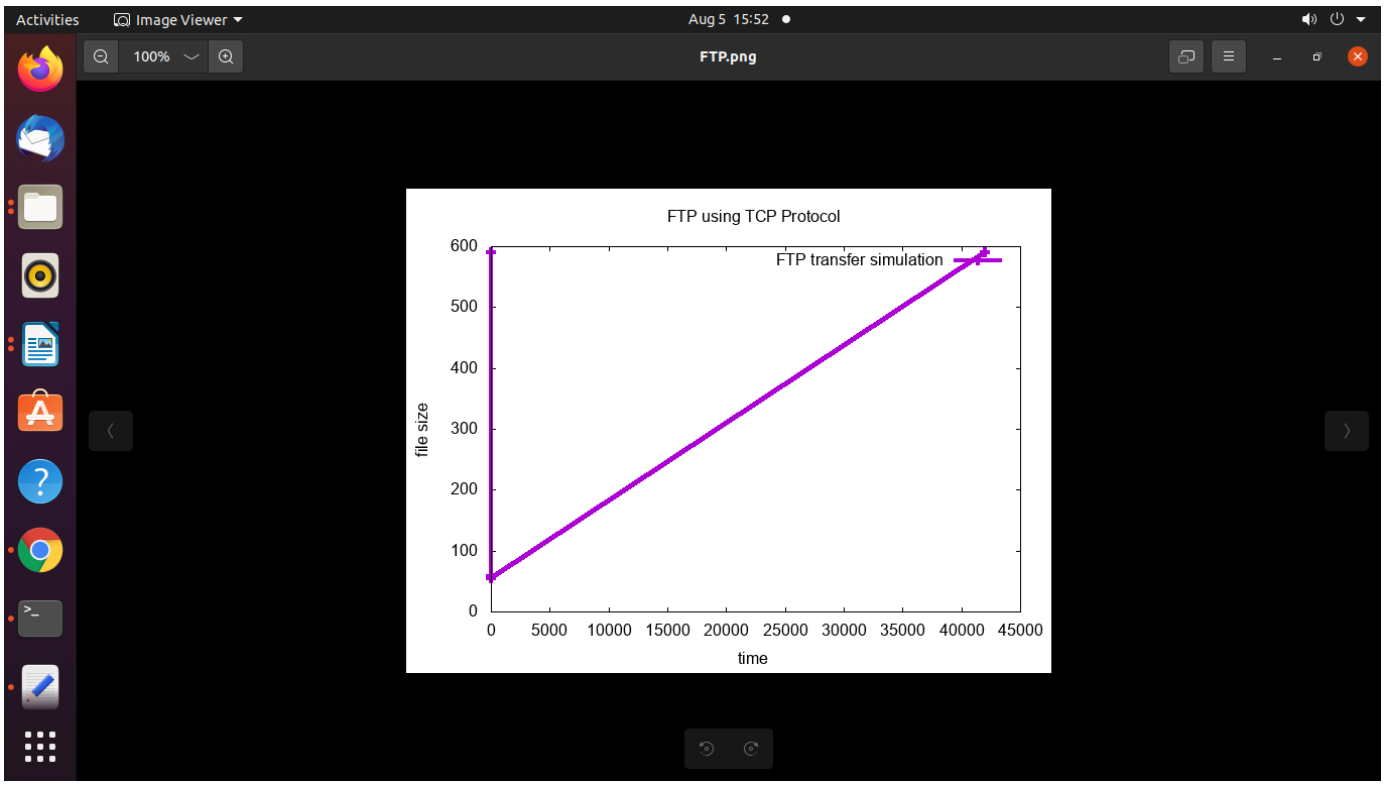
Graph Design for FTP transfer

ftpdemo.txt


Activities
Text Editor
Aug 5 15:51
Open
ftpdemo.txt
~/ns-allinone-3.32/ns-3.32
Save
grid.cc
tcp-large-transfer.cc
ftp.txt
ftpdemo.txt
data.txt
ftp.plt
1 0.000000 10.1.3.1 10.1.2.2 58
2 0.040180 10.1.3.1 10.1.2.2 54
3 0.041080 10.1.3.1 10.1.2.2 590
4 0.041080 10.1.2.2 10.1.3.1 54
5 042024 10.1.3.1 10.1.2.2 590
Plain Text
Tab Width: 8
Ln 5, Col 54
INS

ftp.plt


Activities
Text Editor
Aug 5 15:52
Open
ftp.plt
~/ns-allinone-3.32/ns-3.32
Save
grid.cc
tcp-large-transfer.cc
ftp.txt
ftpdemo.txt
data.txt
ftp.plt
1 set terminal png
2 set output "FTP.png"
3 set title "FTP using TCP Protocol"
4 set xlabel "time"
5 set ylabel "file size"
6 plot "ftpdemo.txt" using 1: 4 with linespoint title "FTP transfer simulation" lw 4
Plain Text
Tab Width: 8
Ln 6, Col 30
INS

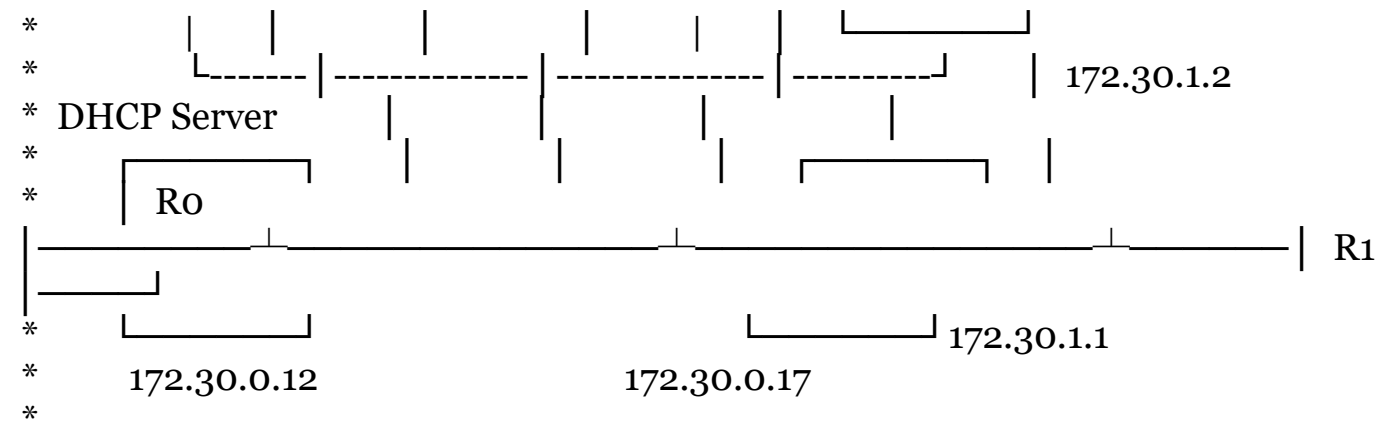

Activities
Image Viewer
Aug 5 15:52
100%
FTP.png
FTP using TCP Protocol
FTP transfer simulation
file size
time
0
100
200
300
400
500
600
0
5000
10000
15000
20000
25000
30000
35000
40000
45000

==========

Program 23 : Program to simulate DHCP server and n clients.

dhcp-server.cc

```
/* -*- 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
|
```



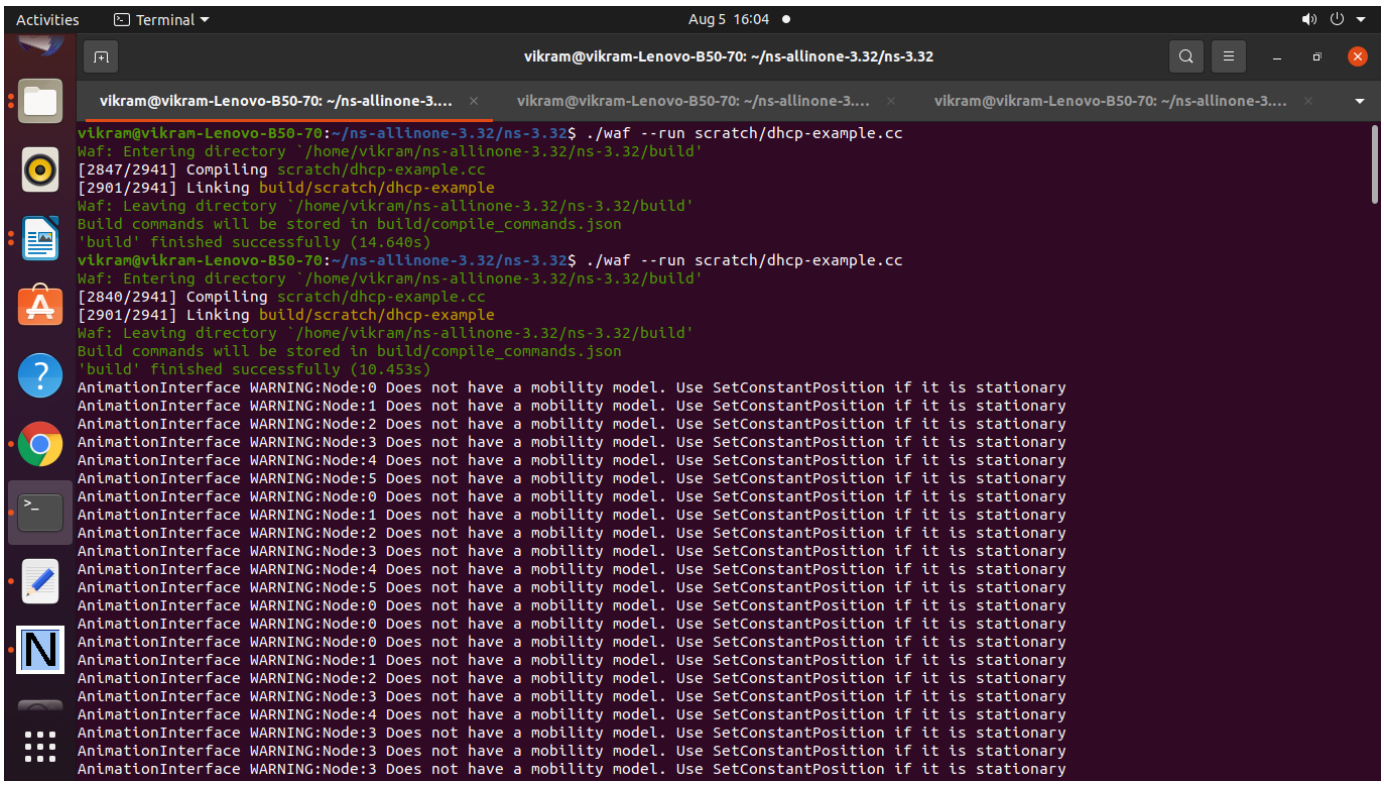

```
 * Things to notice:
 * 1) The routes in A are manually set to have R1 as the default router,
 *    just because using a dynamic outing in this example is an overkill.
 * 2) R1's address is set statically though the DHCP server helper interface.
 *    This is useful to prevent address conflicts with 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"
#include "ns3/netanim-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;
  std::string animFile = "dhcp-server-client-animation.xml";
  cmd.AddValue ("verbose", "turn on the logs", verbose);
  cmd.AddValue ("tracing", "turn on the tracing", tracing);

  cmd.Parse (argc, argv);
```

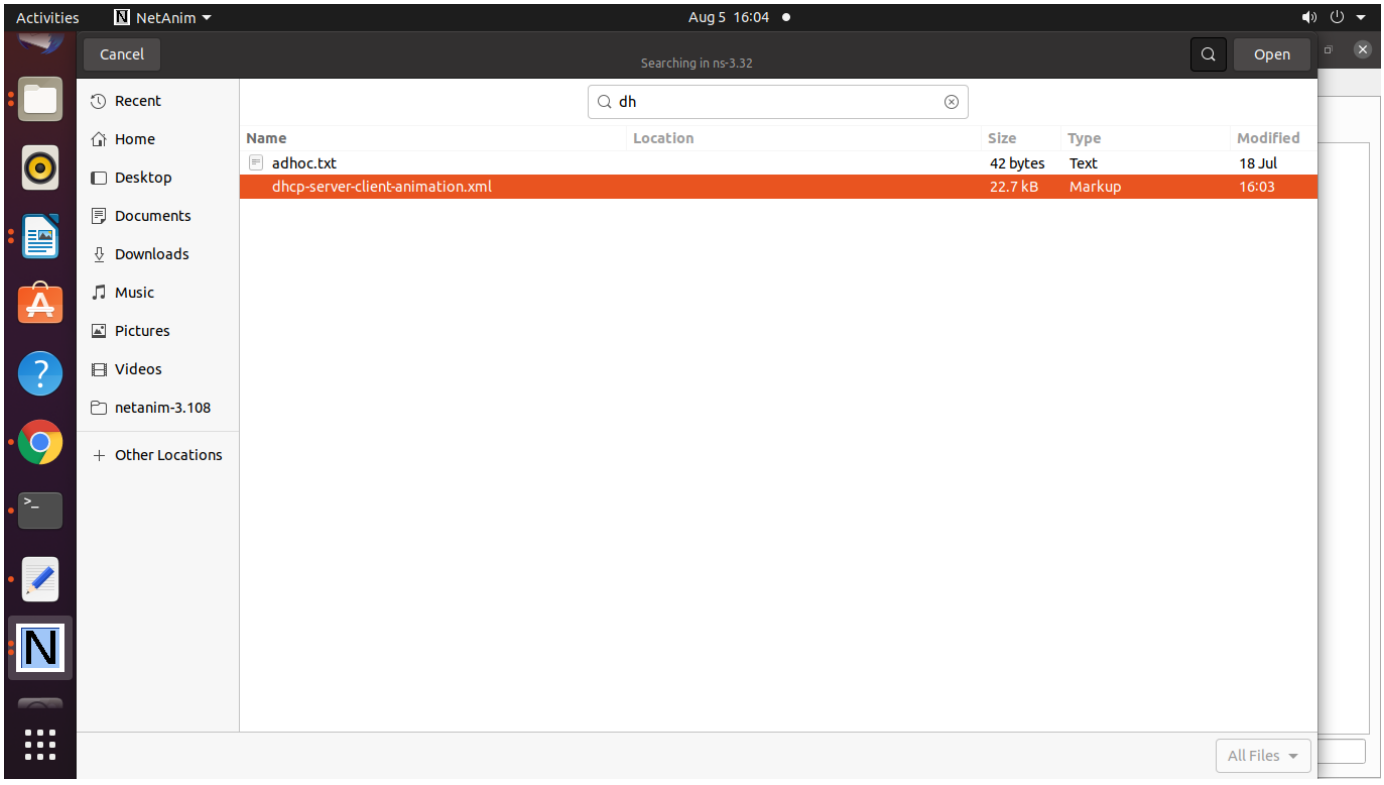
```
// GlobalValue::Bind ("ChecksumEnabled", BooleanValue (true));

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;
```

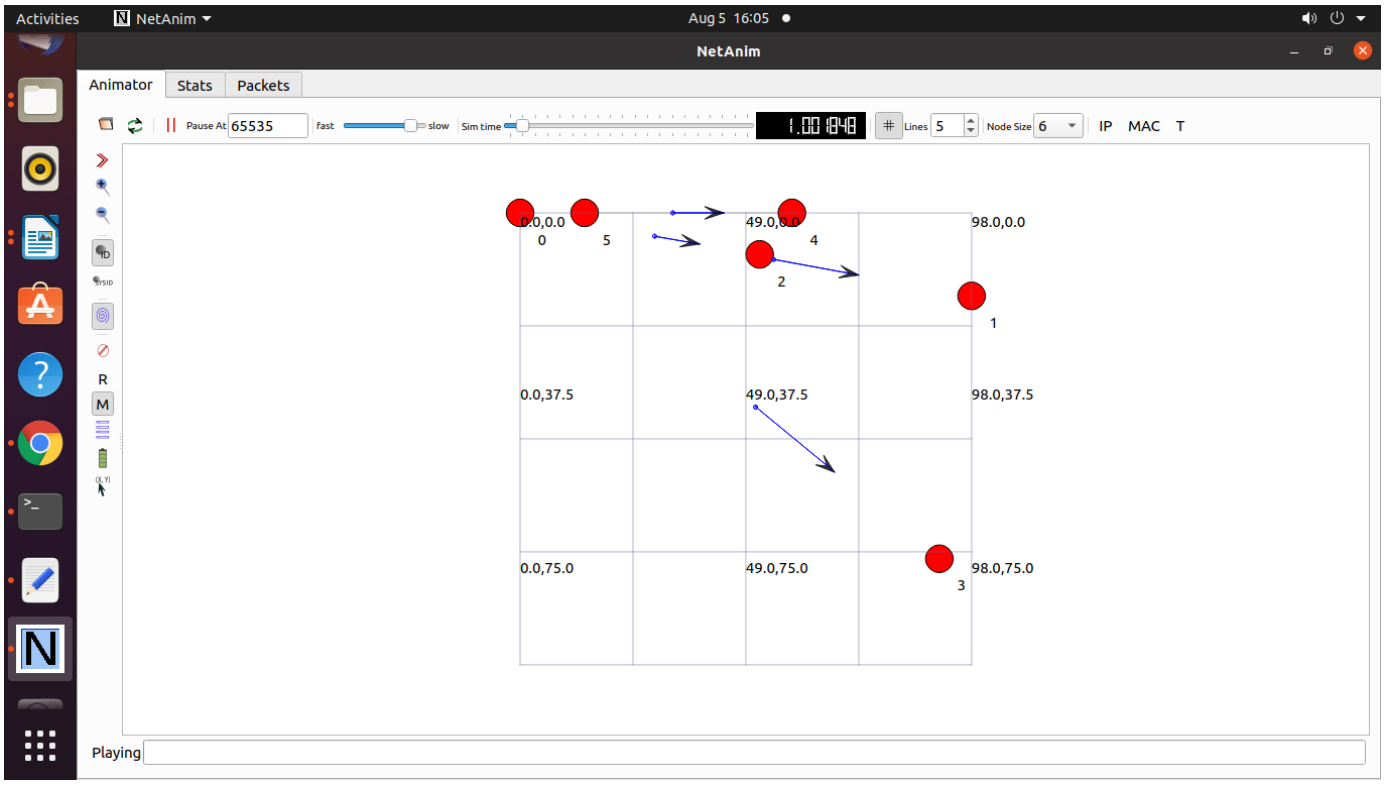
```
  address.SetBase ("172.30.1.0", "255.255.255.0");
  Ipv4InterfaceContainer p2pInterfaces;
  p2pInterfaces = address.Assign (p2pDevices);

  // 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);

  Simulator::Stop (stopTime + Seconds (10.0));
  // Create the animation object and configure for specified output
     AnimationInterface anim (animFile);

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

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

Output:


Activities
Terminal
Aug 5 16:04
vikram@vikram-Lenovo-B50-70: ~/ns-allinone-3.32/ns-3.32
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/dhcp-example.cc
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
[2847/2941] Compiling scratch/dhcp-example.cc
[2901/2941] Linking build/scratch/dhcp-example
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (14.640s)
vikram@vikram-Lenovo-B50-70:~/ns-allinone-3.32/ns-3.32$ ./waf --run scratch/dhcp-example.cc
Waf: Entering directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
[2840/2941] Compiling scratch/dhcp-example.cc
[2901/2941] Linking build/scratch/dhcp-example
Waf: Leaving directory `/home/vikram/ns-allinone-3.32/ns-3.32/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (10.453s)
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:5 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:5 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:0 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:1 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:2 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:4 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary
AnimationInterface WARNING:Node:3 Does not have a mobility model. Use SetConstantPosition if it is stationary


Activities
NetAnim
Aug 5 16:04
Cancel
Searching in ns-3.32
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
dh
Name
Location
Size
Type
Modified
adhoc.txt
42 bytes
Text
18 Jul
dhcp-server-client-animation.xml
22.7 kB
Markup
16:03
All Files


Activities
NetAnim
Aug 5 16:05
NetAnim
Animator
Stats
Packets
Pause At
65535
fast
slow
Sim time
Lines
5
Node Size
6
IP
MAC
T
0.0,0.0
49.0,0.0
98.0,0.0
0.0,37.5
49.0,37.5
98.0,37.5
0.0,75.0
49.0,75.0
98.0,75.0
0
5
4
2
1
3
Playing

Program 23 : Evaluate the performance of the network using flow monitor

third.cc

```
/* -*- 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/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/flow-monitor.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/flow-monitor-module.h"

// Default Network Topology
//
//   Wifi 10.1.3.0
//                 AP
//  *    *    *    *
//  |    |    |    |    10.1.1.0
// n5   n6   n7   n0 -------------- n1   n2   n3   n4
//                   point-to-point  |    |    |    |
//                                   ================
//                                     LAN 10.1.2.0
```

```
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 \"extra\" CSMA nodes/devices", nCsma);
  cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
  cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
  cmd.AddValue ("tracing", "Enable pcap tracing", tracing);

  cmd.Parse (argc,argv);

  // The underlying restriction of 18 is due to the grid position
  // allocator's configuration; the grid layout will exceed the
  // bounding box if more than 18 nodes are provided.
  if (nWifi > 18)
    {
      std::cout << "nWifi should be 18 or less; otherwise grid layout exceeds the
bounding box" << std::endl;
      return 1;
    }

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

  NodeContainer p2pNodes;
  p2pNodes.Create (2);

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

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

```
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;
csmaDevices = csma.Install (csmaNodes);

NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);
NodeContainer wifiApNode = p2pNodes.Get (0);

YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy;
phy.SetChannel (channel.Create ());

WifiHelper wifi;
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");

WifiMacHelper mac;
Ssid ssid = Ssid ("ns-3-ssid");
mac.SetType ("ns3::StaWifiMac",
             "Ssid", SsidValue (ssid),
             "ActiveProbing", BooleanValue (false));

NetDeviceContainer staDevices;
staDevices = wifi.Install (phy, mac, wifiStaNodes);

mac.SetType ("ns3::ApWifiMac",
             "Ssid", SsidValue (ssid));

NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);

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);

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;
p2pInterfaces = address.Assign (p2pDevices);

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

address.SetBase ("10.1.3.0", "255.255.255.0");
address.Assign (staDevices);
address.Assign (apDevices);

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 (wifiStaNodes.Get (nWifi - 1));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));
```

```
  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
Ptr<FlowMonitor> flowMonitor;
FlowMonitorHelper flowHelper;
flowMonitor = flowHelper.InstallAll();
  Simulator::Stop (Seconds (10.0));

  if (tracing)
    {
      phy.SetPcapDataLinkType (WifiPhyHelper::DLT_IEEE802_11_RADIO);
      pointToPoint.EnablePcapAll ("third");
      phy.EnablePcap ("third", apDevices.Get (0));
      csma.EnablePcap ("third", csmaDevices.Get (0), true);
    }

  Simulator::Run ();
  flowMonitor->SerializeToXmlFile("thirdflow.xml",true,true);
  Simulator::Destroy ();
  return 0;
}
```

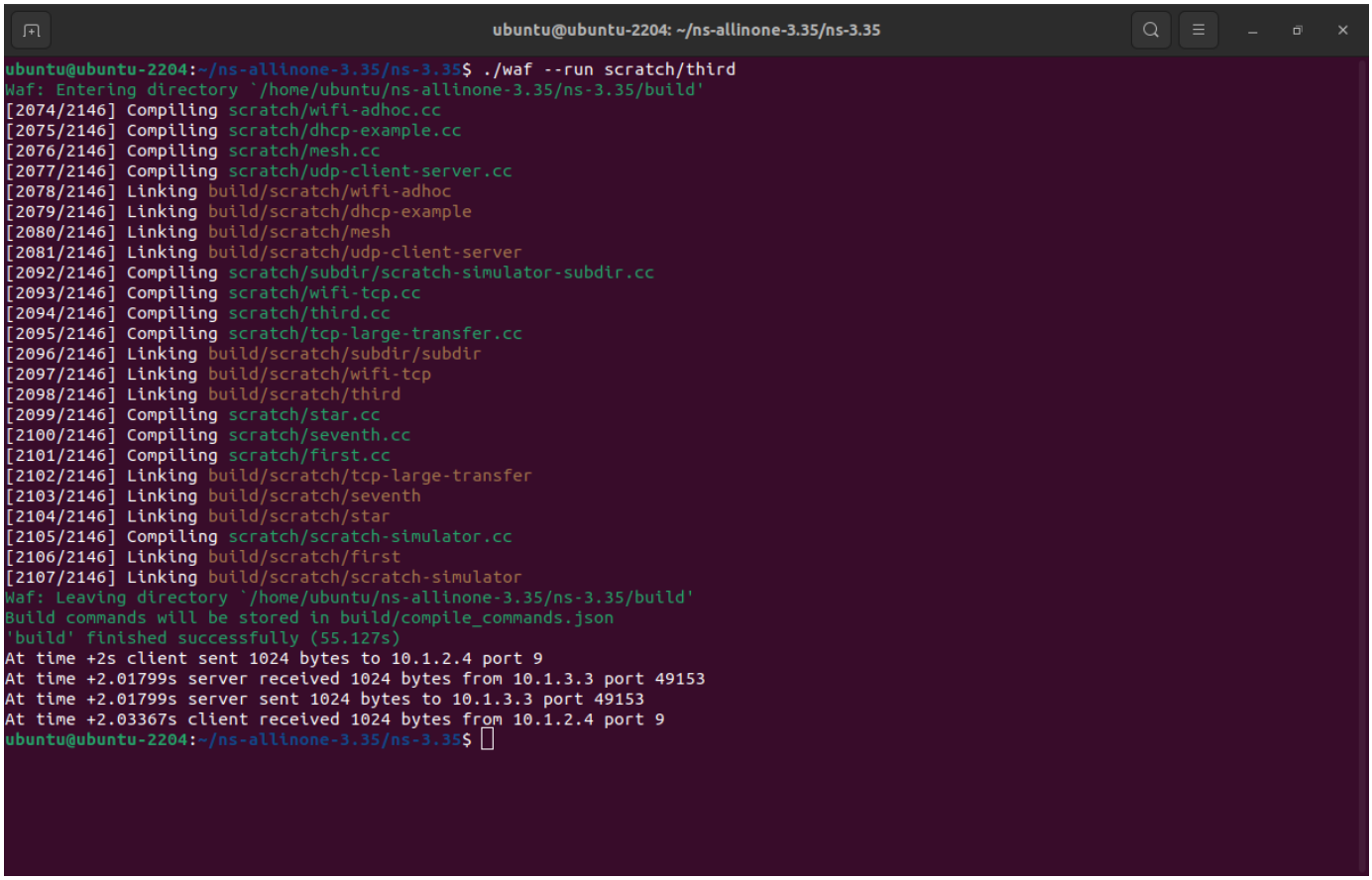
Output

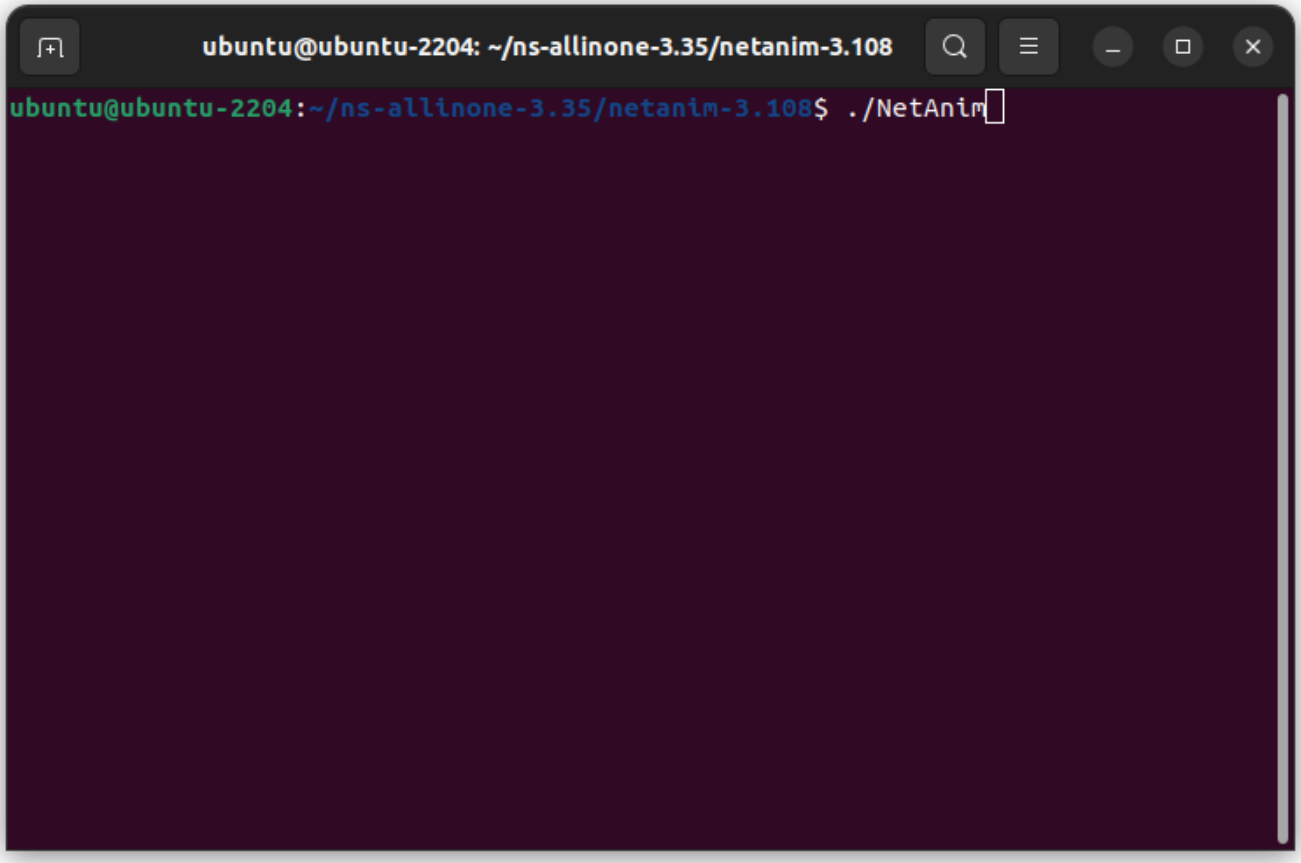
This XML file does not appear to have any style information associated with it. The document tree is shown below.

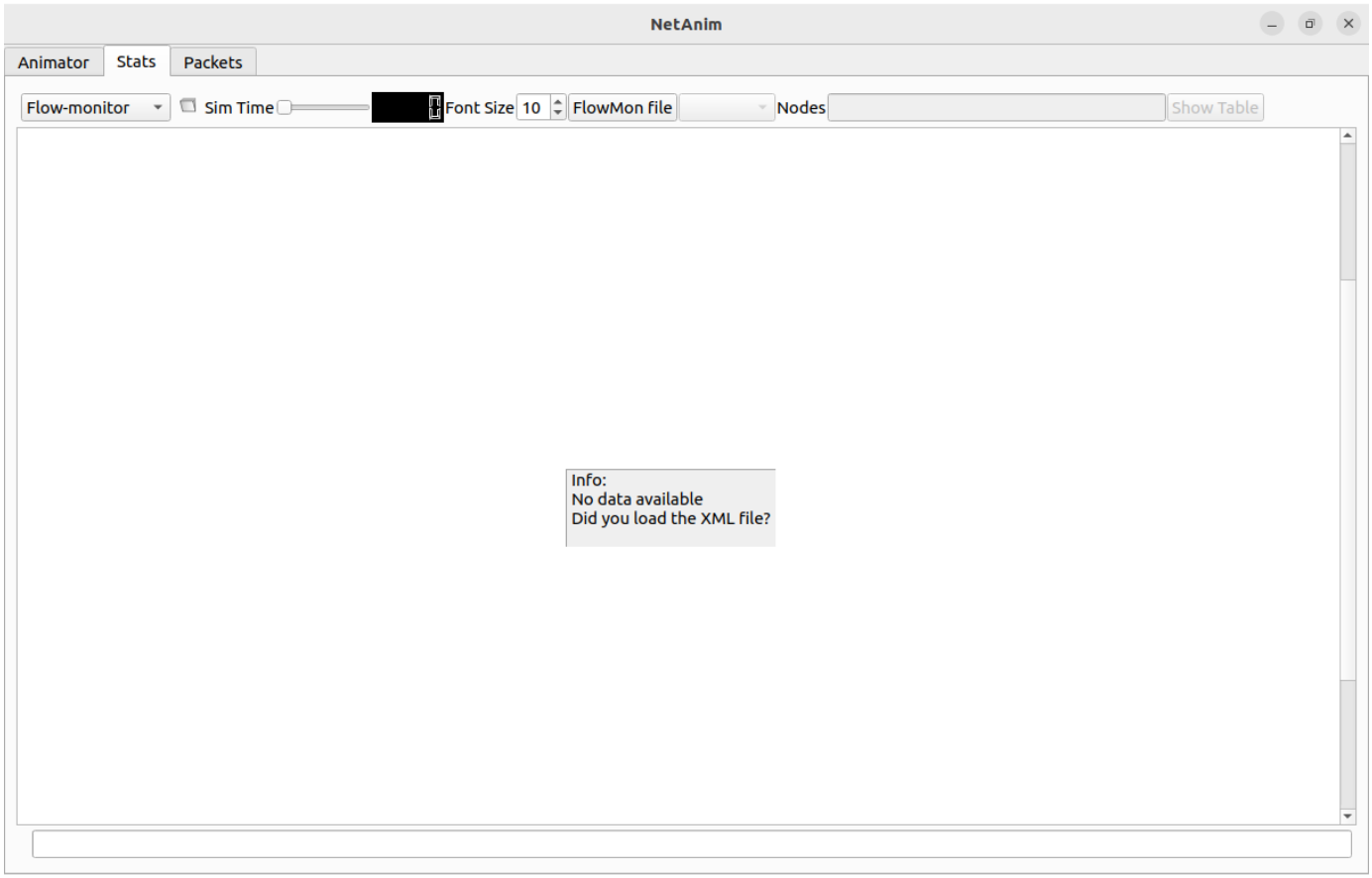
```
<FlowMonitor>
<FlowStats>
<Flow flowId="1" timeFirstTxPacket="+2000000000.0ns" timeFirstRxPacket="+
2012883058.0ns" timeLastTxPacket="+9000000000.0ns" timeLastRxPacket="+9
004770608.0ns" delaySum="+49181295.0ns" jitterSum="+8112464.0ns" lastDelay
="+4770608.0ns" txBytes="8416" rxBytes="8416" txPackets="8" rxPackets="8" lo
stPackets="0" timesForwarded="16">
<delayHistogram nBins="13">
<bin index="4" start="0.004" width="0.001" count="1"/>
<bin index="5" start="0.005" width="0.001" count="6"/>
<bin index="12" start="0.012" width="0.001" count="1"/>
</delayHistogram>
<jitterHistogram nBins="8">
<bin index="0" start="0" width="0.001" count="6"/>
<bin index="7" start="0.007" width="0.001" count="1"/>
```

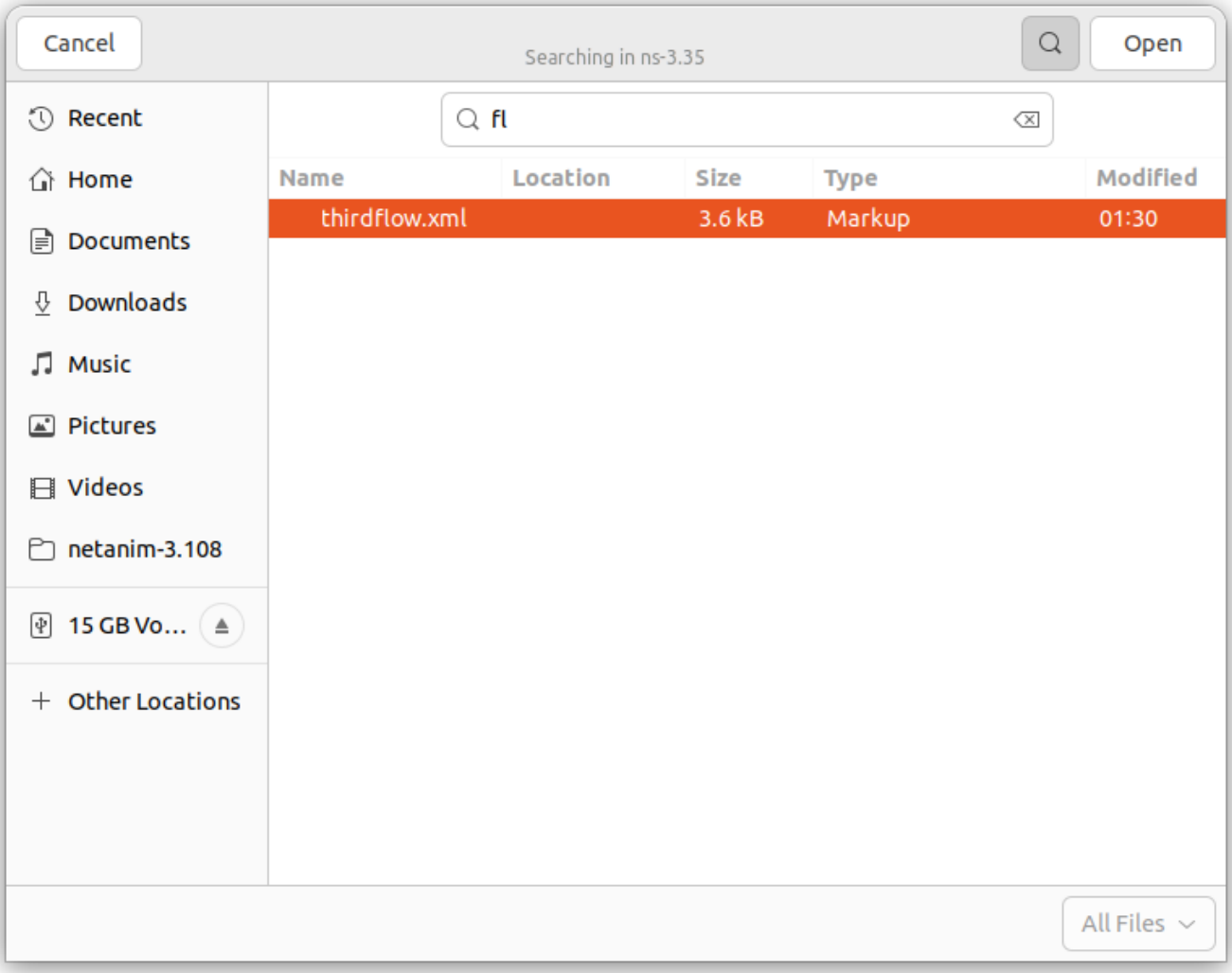
</jitterHistogram>
<packetSizeHistogram nBins="53">
<bin index="52" start="1040" width="20" count="8"/>
</packetSizeHistogram>
<flowInterruptionsHistogram nBins="5">
<bin index="3" start="0.75" width="0.25" count="5"/>
<bin index="4" start="1" width="0.25" count="2"/>
</flowInterruptionsHistogram>
</Flow>
<Flow flowId="2" timeFirstTxPacket="+2012883058.0ns" timeFirstRxPacket="+2021587116.0ns" timeLastTxPacket="+9004770608.0ns" timeLastRxPacket="+9010026216.0ns" delaySum="+45493295.0ns" jitterSum="+3448472.0ns" lastDelay="+5255608.0ns" txBytes="8416" rxBytes="8416" txPackets="8" rxPackets="8" lostPackets="0" timesForwarded="16">
<delayHistogram nBins="9">
<bin index="5" start="0.005" width="0.001" count="7"/>
<bin index="8" start="0.008" width="0.001" count="1"/>
</delayHistogram>
<jitterHistogram nBins="4">
<bin index="0" start="0" width="0.001" count="6"/>
<bin index="3" start="0.003" width="0.001" count="1"/>
</jitterHistogram>
<packetSizeHistogram nBins="53">
<bin index="52" start="1040" width="20" count="8"/>
</packetSizeHistogram>
<flowInterruptionsHistogram nBins="5">
<bin index="3" start="0.75" width="0.25" count="5"/>
<bin index="4" start="1" width="0.25" count="2"/>
</flowInterruptionsHistogram>
</Flow>
</FlowStats>
<Ipv4FlowClassifier>
<Flow flowId="2" sourceAddress="10.1.2.4" destinationAddress="10.1.3.3" protocol="17" sourcePort="9" destinationPort="49153">
<Dscp value="0x0" packets="8"/>
</Flow>
<Flow flowId="1" sourceAddress="10.1.3.3" destinationAddress="10.1.2.4" protocol="17" sourcePort="49153" destinationPort="9">
<Dscp value="0x0" packets="8"/>
</Flow>
</Ipv4FlowClassifier>
<Ipv6FlowClassifier> </Ipv6FlowClassifier>
<FlowProbes>
<FlowProbe index="0">

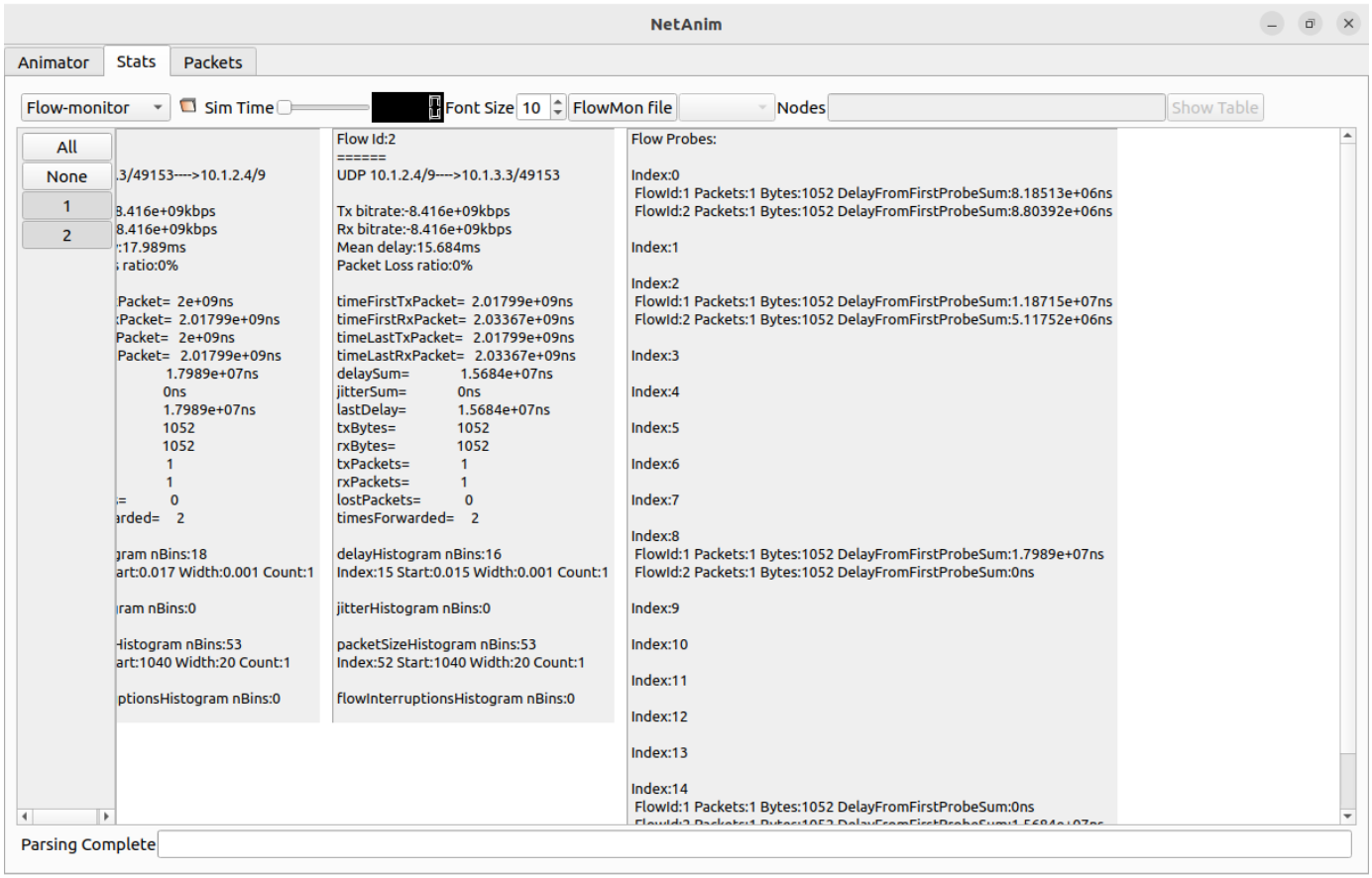
<FlowStats flowId="1" packets="8" bytes="8416" delayFromFirstProbeSum="+179
27455.0ns"> </FlowStats>
<FlowStats flowId="2" packets="8" bytes="8416" delayFromFirstProbeSum="+30
260840.0ns"> </FlowStats>
</FlowProbe>
<FlowProbe index="1"> </FlowProbe>
<FlowProbe index="2">
<FlowStats flowId="1" packets="8" bytes="8416" delayFromFirstProbeSum="+474
18655.0ns"> </FlowStats>
<FlowStats flowId="2" packets="8" bytes="8416" delayFromFirstProbeSum="+76
9640.0ns"> </FlowStats>
</FlowProbe>
<FlowProbe index="3"> </FlowProbe>
<FlowProbe index="4"> </FlowProbe>
<FlowProbe index="5"> </FlowProbe>
<FlowProbe index="6"> </FlowProbe>
<FlowProbe index="7"> </FlowProbe>
<FlowProbe index="8">
<FlowStats flowId="1" packets="8" bytes="8416" delayFromFirstProbeSum="+491
81295.0ns"> </FlowStats>
<FlowStats flowId="2" packets="8" bytes="8416" delayFromFirstProbeSum="+0.0
ns"> </FlowStats>
</FlowProbe>
<FlowProbe index="9"> </FlowProbe>
<FlowProbe index="10"> </FlowProbe>
<FlowProbe index="11"> </FlowProbe>
<FlowProbe index="12"> </FlowProbe>
<FlowProbe index="13"> </FlowProbe>
<FlowProbe index="14">
<FlowStats flowId="1" packets="8" bytes="8416" delayFromFirstProbeSum="+0.0
ns"> </FlowStats>
<FlowStats flowId="2" packets="8" bytes="8416" delayFromFirstProbeSum="+45
493295.0ns"> </FlowStats>
</FlowProbe>
<FlowProbe index="15"> </FlowProbe>
</FlowProbes>
</FlowMonitor>


ubuntu@ubuntu-2204: ~/ns-allinone-3.35/ns-3.35
ubuntu@ubuntu-2204:~/ns-allinone-3.35/ns-3.35$ ./waf --run scratch/third
Waf: Entering directory `/home/ubuntu/ns-allinone-3.35/ns-3.35/build'
[2074/2146] Compiling scratch/wifi-adhoc.cc
[2075/2146] Compiling scratch/dhcp-example.cc
[2076/2146] Compiling scratch/mesh.cc
[2077/2146] Compiling scratch/udp-client-server.cc
[2078/2146] Linking build/scratch/wifi-adhoc
[2079/2146] Linking build/scratch/dhcp-example
[2080/2146] Linking build/scratch/mesh
[2081/2146] Linking build/scratch/udp-client-server
[2092/2146] Compiling scratch/subdir/scratch-simulator-subdir.cc
[2093/2146] Compiling scratch/wifi-tcp.cc
[2094/2146] Compiling scratch/third.cc
[2095/2146] Compiling scratch/tcp-large-transfer.cc
[2096/2146] Linking build/scratch/subdir/subdir
[2097/2146] Linking build/scratch/wifi-tcp
[2098/2146] Linking build/scratch/third
[2099/2146] Compiling scratch/star.cc
[2100/2146] Compiling scratch/seventh.cc
[2101/2146] Compiling scratch/first.cc
[2102/2146] Linking build/scratch/tcp-large-transfer
[2103/2146] Linking build/scratch/seventh
[2104/2146] Linking build/scratch/star
[2105/2146] Compiling scratch/scratch-simulator.cc
[2106/2146] Linking build/scratch/first
[2107/2146] Linking build/scratch/scratch-simulator
Waf: Leaving directory `/home/ubuntu/ns-allinone-3.35/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (55.127s)
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
At time +2.01799s server received 1024 bytes from 10.1.3.3 port 49153
At time +2.01799s server sent 1024 bytes to 10.1.3.3 port 49153
At time +2.03367s client received 1024 bytes from 10.1.2.4 port 9
ubuntu@ubuntu-2204:~/ns-allinone-3.35/ns-3.35$


ubuntu@ubuntu-2204: ~/ns-allinone-3.35/netanim-3.108
ubuntu@ubuntu-2204:~/ns-allinone-3.35/netanim-3.108$ ./NetAnim


NetAnim
Animator
Stats
Packets
Flow-monitor
Sim Time
Font Size
10
FlowMon file
Nodes
Show Table
Info:
No data available
Did you load the XML file?


Cancel
Searching in ns-3.35
Open
Recent
Home
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
15 GB Vo…
Other Locations
fl
Name
Location
Size
Type
Modified
thirdflow.xml
3.6 kB
Markup
01:30
All Files


NetAnim
Animator
Stats
Packets
Flow-monitor
Sim Time
Font Size
10
FlowMon file
Nodes
Show Table
All
None
1
2
.3/49153---->10.1.2.4/9
8.416e+09kbps
8.416e+09kbps
:17.989ms
ratio:0%
Packet= 2e+09ns
Packet= 2.01799e+09ns
Packet= 2e+09ns
Packet= 2.01799e+09ns
1.7989e+07ns
0ns
1.7989e+07ns
1052
1052
1
1
= 0
rded= 2
gram nBins:18
art:0.017 Width:0.001 Count:1
ram nBins:0
Histogram nBins:53
art:1040 Width:20 Count:1
ptionsHistogram nBins:0
Flow Id:2
======
UDP 10.1.2.4/9---->10.1.3.3/49153
Tx bitrate:-8.416e+09kbps
Rx bitrate:-8.416e+09kbps
Mean delay:15.684ms
Packet Loss ratio:0%
timeFirstTxPacket= 2.01799e+09ns
timeFirstRxPacket= 2.03367e+09ns
timeLastTxPacket= 2.01799e+09ns
timeLastRxPacket= 2.03367e+09ns
delaySum= 1.5684e+07ns
jitterSum= 0ns
lastDelay= 1.5684e+07ns
txBytes= 1052
rxBytes= 1052
txPackets= 1
rxPackets= 1
lostPackets= 0
timesForwarded= 2
delayHistogram nBins:16
Index:15 Start:0.015 Width:0.001 Count:1
jitterHistogram nBins:0
packetSizeHistogram nBins:53
Index:52 Start:1040 Width:20 Count:1
flowInterruptionsHistogram nBins:0
Flow Probes:
Index:0
FlowId:1 Packets:1 Bytes:1052 DelayFromFirstProbeSum:8.18513e+06ns
FlowId:2 Packets:1 Bytes:1052 DelayFromFirstProbeSum:8.80392e+06ns
Index:1
Index:2
FlowId:1 Packets:1 Bytes:1052 DelayFromFirstProbeSum:1.18715e+07ns
FlowId:2 Packets:1 Bytes:1052 DelayFromFirstProbeSum:5.11752e+06ns
Index:3
Index:4
Index:5
Index:6
Index:7
Index:8
FlowId:1 Packets:1 Bytes:1052 DelayFromFirstProbeSum:1.7989e+07ns
FlowId:2 Packets:1 Bytes:1052 DelayFromFirstProbeSum:0ns
Index:9
Index:10
Index:11
Index:12
Index:13
Index:14
FlowId:1 Packets:1 Bytes:1052 DelayFromFirstProbeSum:0ns
Parsing Complete

Program 24: Congestion Control

seventh.cc

```
/* -*- 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"
#include "ns3/stats-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SeventhScriptExample");

//
==========================================================
=======================
//
//       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.
//
// NOTE: If this example gets modified, do not forget to update the .png figure
// in src/stats/docs/seventh-packet-byte-count.png
//
//============================================================================
//
class MyApp : public Application
{
```

```
public:
  MyApp ();
  virtual ~MyApp ();

  /**
   * Register this type.
   * \return The TypeId.
   */
  static TypeId GetTypeId (void);
  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;
```

```
}

/* static */
TypeId MyApp::GetTypeId (void)
{
  static TypeId tid = TypeId ("MyApp")
    .SetParent<Application> ()
    .SetGroupName ("Tutorial")
    .AddConstructor<MyApp> ()
    ;
  return tid;
}

void
MyApp::Setup (Ptr<Socket> socket, Address address, uint32_t pack-
etSize, 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;
  if (InetSocketAddress::IsMatchingType (m_peer))
    {
      m_socket->Bind ();
    }
  else
    {
      m_socket->Bind6 ();
    }
  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 (Ptr<OutputStreamWrapper> stream, uint32_t oldCwnd,
uint32_t newCwnd)
{
  NS_LOG_UNCOND (Simulator::Now ().GetSeconds () << "\t" << new-
Cwnd);
  *stream->GetStream () << Simulator::Now ().GetSeconds () << "\t" <<
oldCwnd << "\t" << newCwnd << std::endl;
```

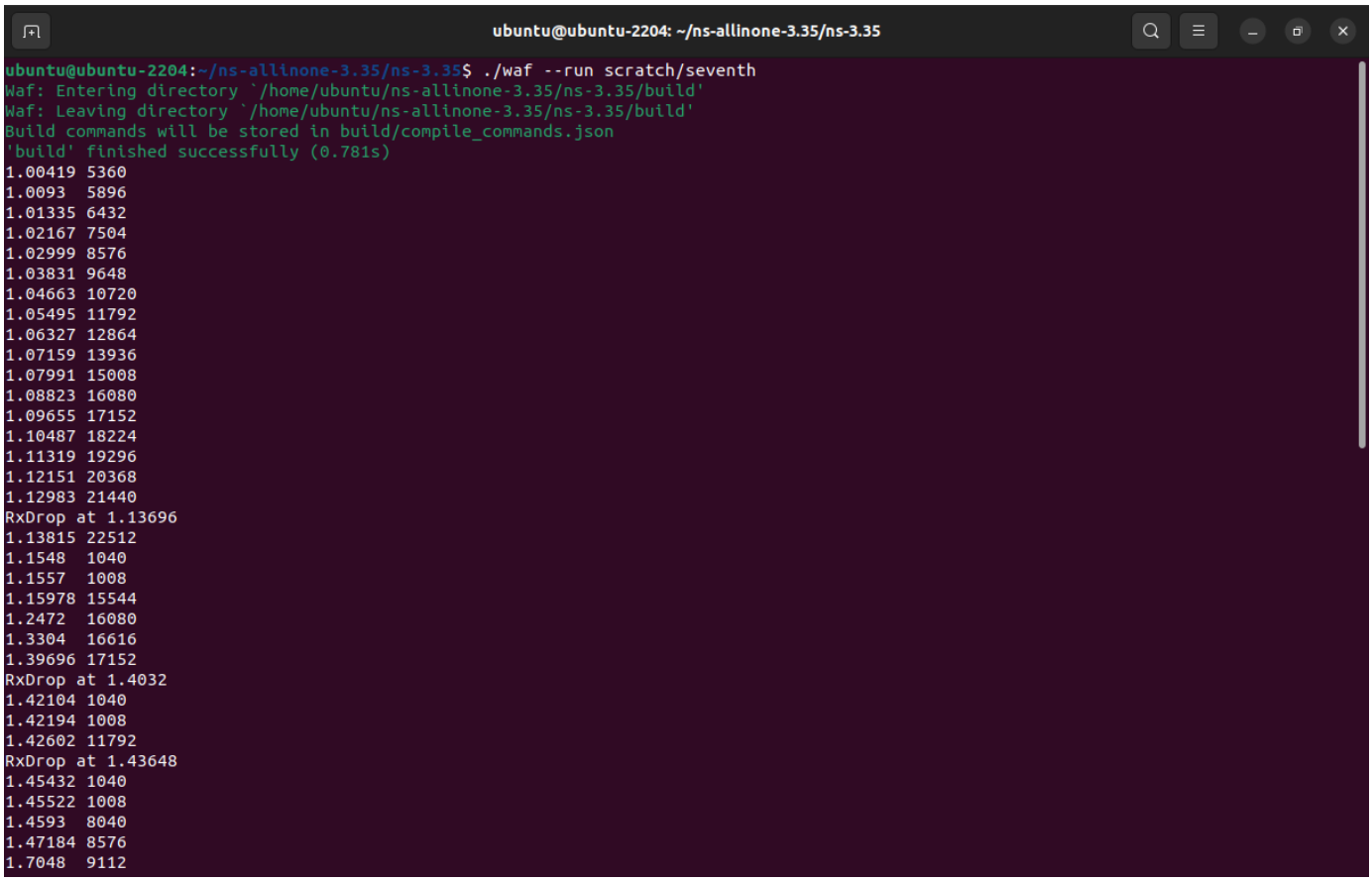
```
}

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

int
main (int argc, char *argv[])
{
  bool useV6 = false;

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

  uint16_t sinkPort = 8080;
  Address sinkAddress;
  Address anyAddress;
  std::string probeType;
  std::string tracePath;
  if (useV6 == false)
```

```
  {
    Ipv4AddressHelper address;
    address.SetBase ("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer interfaces = address.Assign (devices);
    sinkAddress = InetSocketAddress (interfaces.GetAddress (1), sink-
Port);
    anyAddress = InetSocketAddress (Ipv4Address::GetAny (), sink-
Port);
    probeType = "ns3::Ipv4PacketProbe";
    tracePath = "/NodeList/*/$ns3::Ipv4L3Protocol/Tx";
  }
 else
  {
    Ipv6AddressHelper address;
    address.SetBase ("2001:0000:food:cafe::", Ipv6Prefix (64));
    Ipv6InterfaceContainer interfaces = address.Assign (devices);
    sinkAddress = Inet6SocketAddress (interfaces.GetAddress (1,1),
sinkPort);
    anyAddress = Inet6SocketAddress (Ipv6Address::GetAny (), sink-
Port);
    probeType = "ns3::Ipv6PacketProbe";
    tracePath = "/NodeList/*/$ns3::Ipv6L3Protocol/Tx";
  }

 PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory", an-
yAddress);
 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 ());

 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.));

 AsciiTraceHelper asciiTraceHelper;
 Ptr<OutputStreamWrapper> stream = asciiTraceHelper.Create-
FileStream ("seventh.cwnd");
```

```
  ns3TcpSocket->TraceConnectWithoutContext ("CongestionWindow",
MakeBoundCallback (&CwndChange, stream));

  PcapHelper pcapHelper;
  Ptr<PcapFileWrapper> file = pcapHelper.CreateFile ("seventh.pcap",
std::ios::out, PcapHelper::DLT_PPP);
  devices.Get (1)->TraceConnectWithoutContext ("PhyRxDrop", Make-
BoundCallback (&RxDrop, file));

  // Use GnuplotHelper to plot the packet byte count over time
  GnuplotHelper plotHelper;

  // Configure the plot.  The first argument is the file name prefix
  // for the output files generated.  The second, third, and fourth
  // arguments are, respectively, the plot title, x-axis, and y-axis labels
  plotHelper.ConfigurePlot ("seventh-packet-byte-count",
                            "Packet Byte Count vs. Time",
                            "Time (Seconds)",
                            "Packet Byte Count");

  // Specify the probe type, trace source path (in configuration
namespace), and
  // probe output trace source ("OutputBytes") to plot.  The fourth argu-
ment
  // specifies the name of the data series label on the plot.  The last
  // argument formats the plot by specifying where the key should be
placed.
  plotHelper.PlotProbe (probeType,
                        tracePath,
                        "OutputBytes",
                        "Packet Byte Count",
                        GnuplotAggregator::KEY_BELOW);

  // Use FileHelper to write out the packet byte count over time
  FileHelper fileHelper;

  // Configure the file to be written, and the formatting of output data.
  fileHelper.ConfigureFile ("seventh-packet-byte-count",
                            FileAggregator::FORMATTED);

  // Set the labels for this formatted output file.
  fileHelper.Set2dFormat ("Time (Seconds) = %.3e\tPacket Byte Count
= %.0f");
```

```
  // Specify the probe type, trace source path (in configuration
namespace), and
  // probe output trace source ("OutputBytes") to write.
  fileHelper.WriteProbe (probeType,
                         tracePath,
                         "OutputBytes");

  Simulator::Stop (Seconds (20));
  Simulator::Run ();
  Simulator::Destroy ();

  return 0;
}
```

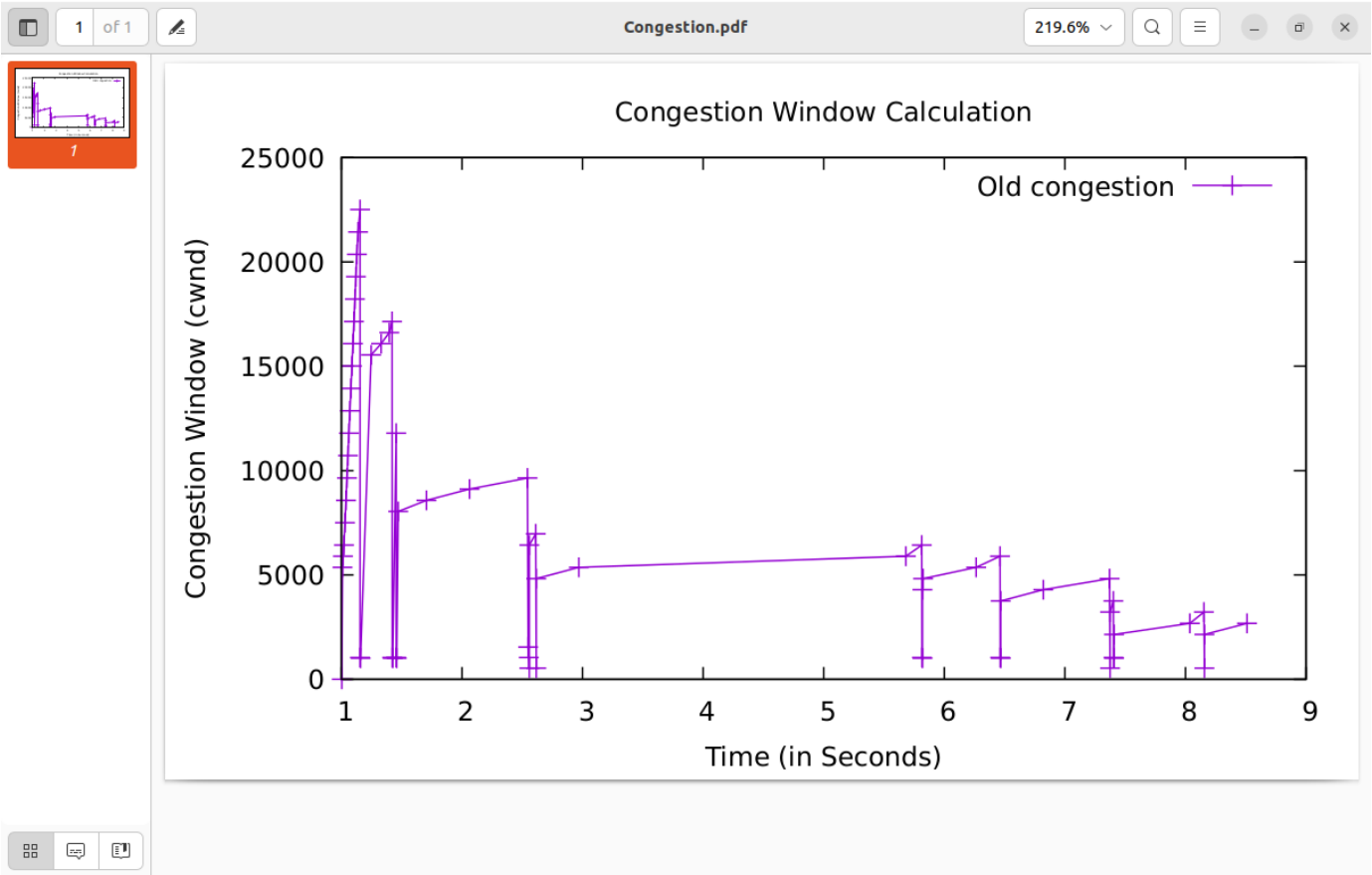
Output:



congestion.plt

```
set terminal pdf
set output "Congestion1.pdf"
```

set title "Congestion Window Calculation"
set xlabel "Time (in Seconds)"
set ylabel "Congestion Window (cwnd)"
plot "seventh.cwnd" using 1:3 with linespoint title "Old congestion"



Prog. 25 :

Monitoring and Analyzing the Network Traffic

- Analysis of Protocols using Wireshark
- HTTP, HTTPS, DNS
- TCP, UDP
- IP, ICMP
- ARP
- DHCP

HTML Documents with Embedded Objects

Now that we've seen how Wireshark displays the captured packet traffic for large HTML files, we can look at what happens when your browser downloads a file with embedded objects, i.e., a file that includes other objects (in the example below, image files) that are stored on another server(s).

Do the following:

- Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
- Start up the Wireshark packet sniffer

Enter the following URL into your browser
http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html

. The URL
http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html is password protected. The username is "wireshark-students" (without the quotes), and the password is "network" (again, without the quotes). So let's access this "secure" password-protected site. Do the following:

- Make sure your browser's cache is cleared, as discussed above, and close down your browser. Then, start up your browser
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser
  http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html
  Type the requested user name and password into the pop up box.
- Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

Note: If you are unable to run Wireshark on a live network connection, you can use the "classic" http-ethereal-trace-5 packet trace, or other additional traces, as notes in footnote 2, to answer the questions below

DNS

Tracing DNS with Wireshark

Now that we are familiar with nslookup and clearing the DNS resolver cache, we're ready to get down to some serious business. Let's first capture the DNS messages that are generated by ordinary Web-surfing activity.

- Clear the DNS cache in your host, as described above.
- Open your Web browser and clear your browser cache.
- Open Wireshark and enter ip.addr == <your_IP_address> into the display filter, where <your_IP_address> is the IPv4 address of your computer[1]. With this filter, Wireshark will only display packets that either originate from, or are destined to, your host.
- Start packet capture in Wireshark.
- With your browser, visit the Web page: http://gaia.cs.umass.edu/kurose_ross/
- Stop packet capture.

If you are unable to run Wireshark on a live network connection, you can download a packet trace file that was captured while following the steps above on one of the author's computers[2]. Answer the following questions.

5. Locate the first DNS query message resolving the name gaia.cs.umass.edu. What is the packet number[3] in the trace for the DNS query message? Is this query message sent over UDP or TCP?
6. Now locate the corresponding DNS response to the initial DNS query. What is the packet number in the trace for the DNS response message? Is this response message received via UDP or TCP?
7. What is the destination port for the DNS query message? What is the source port of the DNS response message?
8. To what IP address is the DNS query message sent?
9. Examine the DNS query message. How many "questions" does this DNS message contain? How many "answers" answers does it contain?

---

[1] If you're not sure how to find the IP address of your computer, you can search the Web for articles for your operating system. Windows 10 info is here; Mac info is here; Linux info is here

[2] You can download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip and extract the trace file dns-wireshark-trace1-1. These trace files can be used to answer these Wireshark lab questions without actually capturing packets on your own. Each trace was made using Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you've downloaded a trace file, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the trace file name.

[3] Remember that this "packet number" is assigned by Wireshark for listing purposes only; it is NOT a packet number contained in any real packet header.

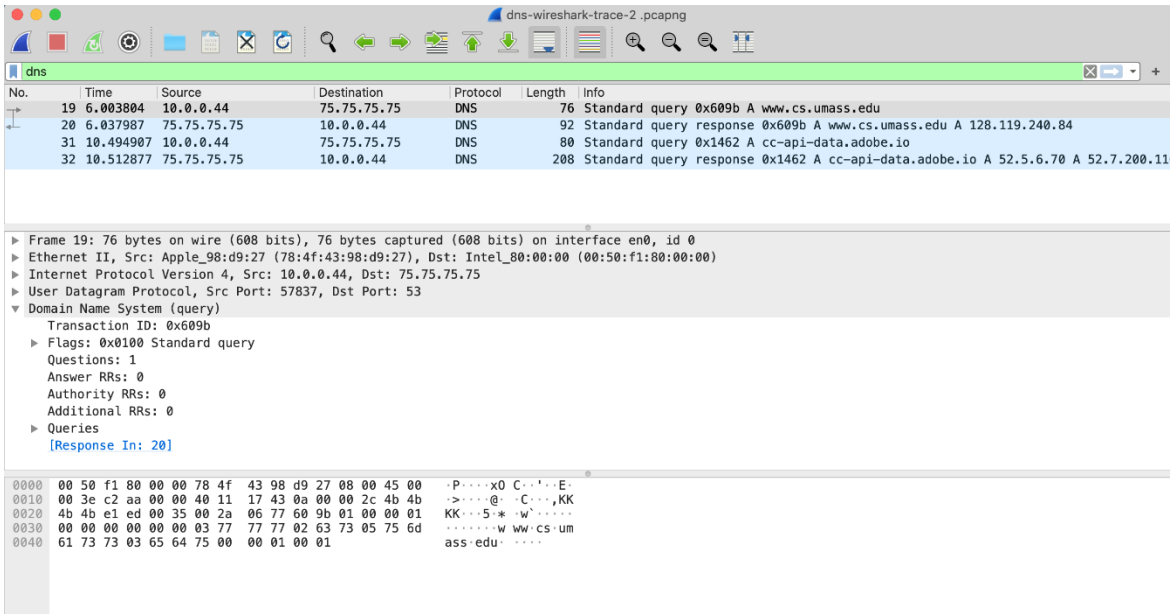
10. Examine the DNS response message to the initial query message. How many "questions" does this DNS message contain? How many "answers" answers does it contain?

11. The web page for the base file http://gaia.cs.umass.edu/kurose_ross/ references the image object http://gaia.cs.umass.edu/kurose_ross/header_graphic_book_8E_2.jpg , which, like the base webpage, is on gaia.cs.umass.edu. What is the packet number in the trace for the initial HTTP GET request for the base file http://gaia.cs.umass.edu/kurose_ross/? What is the packet number in the trace of the DNS query made to resolve gaia.cs.umass.edu so that this initial HTTP request can be sent to the gaia.cs.umass.edu IP address? What is the packet number in the trace of the received DNS response? What is the packet number in the trace for the HTTP GET request for the image object http://gaia.cs.umass.edu/kurose_ross/header_graphic_book_8E2.jpg? What is the packet number in the DNS query made to resolve gaia.cs.umass.edu so that this second HTTP request can be sent to the gaia.cs.umass.edu IP address? Discuss how DNS caching affects the answer to this last question.

Now let's play with nslookup[4].

- Start packet capture.
- Do an nslookup on www.cs.umass.edu
- Stop packet capture.

You should get a trace that looks something like the following in your Wireshark window. Let's look at the first type A query (which is packet number 19 in the figure below, and indicated by the "A" in the *Info* column for that packet.



[4] If you are unable to run Wireshark and capture a trace file, or are using an LMS, use the trace file dns-wireshark-trace-2 in the zip file of traces in the footnote above to answer questions 12-16 below.

12. What is the destination port for the DNS query message? What is the source port of the DNS response message?
13. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?
14. Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?
15. Examine the DNS response message to the query message. How many "questions" does this DNS response message contain? How many "answers"?

Last, let's use nslookup to issue a command that will return a type NS DNS record, Enter the following command:
nslookup –type=NS umass.edu
and then answer the following questions[5] :

16. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?
17. Examine the DNS query message. How many questions does the query have? Does the query message contain any "answers"?
18. Examine the DNS response message (in particular the DNS response message that has type "NS"). How many answers does the response have? What information is contained in the answers? How many additional resource records are returned? What additional information is included in these additional resource records (if additional information is returned)?

TCP
Capturing a bulk TCP transfer from your computer to a remote server

Before beginning our exploration of TCP, we'll need to use Wireshark to obtain a packet trace of the TCP transfer of a file from your computer to a remote server. You'll do so by accessing a Web page that will allow you to enter the name of a file stored on your computer (which contains the ASCII text of *Alice in Wonderland*), and then transfer the file to a Web server using the HTTP POST method (see section 2.2.3 in the text). We're using the POST method rather than the GET method as we'd like to transfer a large amount of data *from* your computer to another computer. Of course, we'll be running Wireshark during this time to obtain the trace of the TCP segments sent and received from your computer.

Do the following:

[5] If you are unable to run Wireshark and capture a trace file, or are using an LMS, use the trace file *dns-wireshark-trace-3* in the zip file of traces in the footnote above to answer questions 17-19 below.

- Start up your web browser. Go the http://gaia.cs.umass.edu/wireshark-labs/alice.txt and retrieve an ASCII copy of *Alice in Wonderland*. Store this as a .txt file somewhere on your computer.
- Next go to http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html.
- You should see a screen that looks like Figure 1.

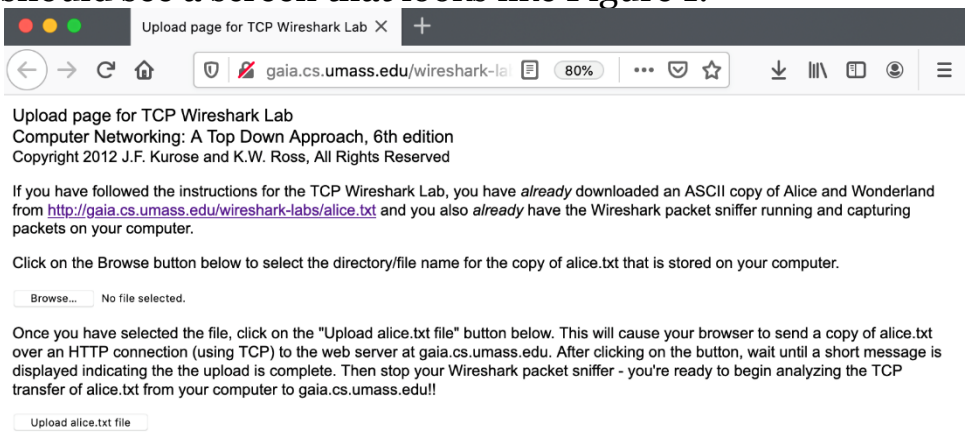


Figure 1: Page to upload the alice.txt file from your computer to gaia.cs.umass.edu

- Use the *Browse* button in this form to the file on your computer that you just created containing *Alice in Wonderland*. Don't press the "*Upload alice.txt file*" button yet.
- Now start up Wireshark and begin packet capture (see the earlier Wireshark labs if you need a refresher on how to do this).
- Returning to your browser, press the "*Upload alice.txt file*" button to upload the file to the gaia.cs.umass.edu server. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.
- Stop Wireshark packet capture. Your Wireshark window should look similar to the window shown in Figure 2.

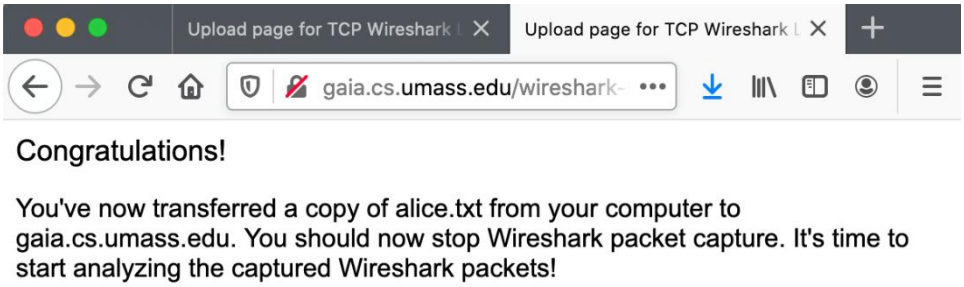


Figure 2: Success! You've uploaded a file to gaia.cs.umass.edu and have hopefully captured a Wireshark packet trace while doing so.

If you are unable to run Wireshark on a live network connection, you can download a packet trace that was captured while following the steps above on one of the author's computers [6]. In addition, you may well find it valuable to download this trace even if you've captured your own trace and use it, as well as your own trace, when you explore the questions below.

2. A first look at the captured trace

Before analyzing the behavior of the TCP connection in detail, let's take a high-level view of the trace.

Let's start by looking at the HTTP POST message that uploaded the alice.txt file to gaia.cs.umass.edu from your computer. Find that file in your Wireshark trace, and expand the HTTP message so we can take a look at the HTTP POST message more carefully. Your Wireshark screen should look something like Figure 3.

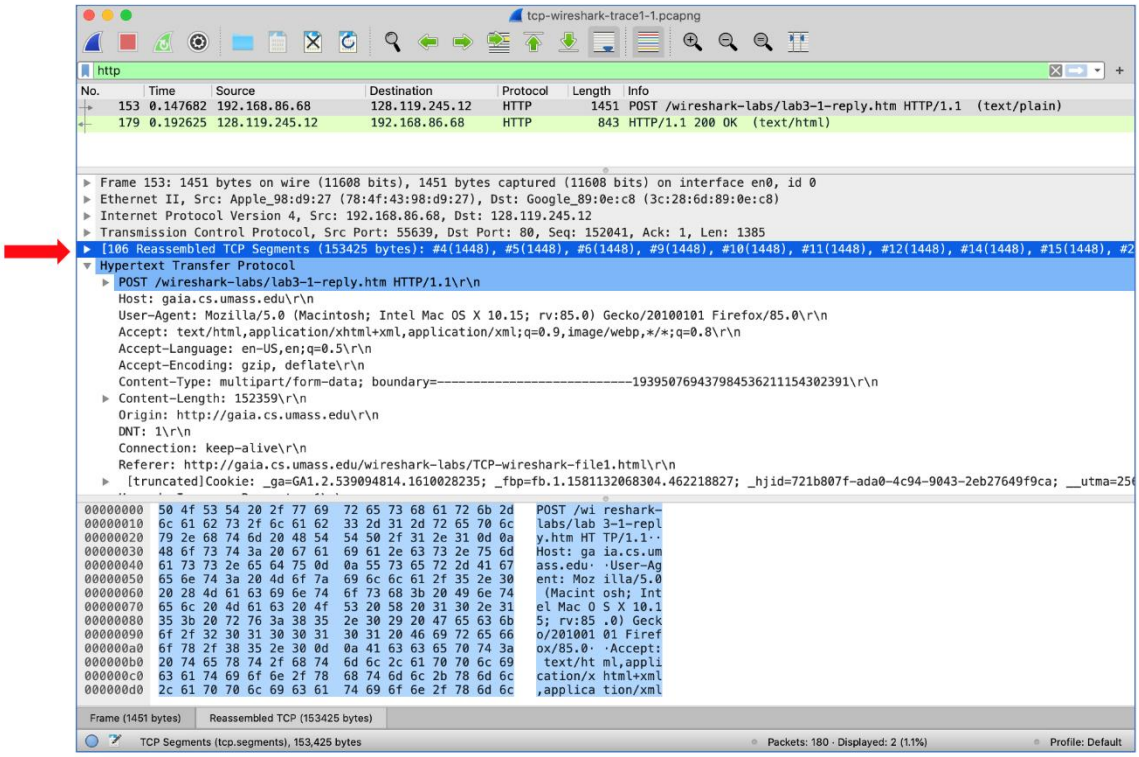


Figure 3: expanding the HTTP POST message that uploaded alice.txt from your computer to gaia.cs.umass.edu

There are a few things to note here:

[6] You can download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip and extract the trace file *tcp-wireshark-trace1-1*. This trace file can be used to answer this Wireshark lab without actually capturing packets on your own. This trace was made using Wireshark running on one of the author's computers, while performing the steps indicated in this Wireshark lab. Once you've downloaded a trace file, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the trace file name.

- The body of your application-layer HTTP POST message contains the contents of the file alice.txt, which is a large file of more than 152K bytes. OK – it's not *that* large, but it's going to be too large for this one HTTP POST message to be contained in just one TCP segment!
- In fact, as shown in the Wireshark window in Figure 3 we see that the HTTP POST message was spread across 106 TCP segments. This is shown where the red arrow is placed in Figure 3 [Aside: Wireshark doesn't have a red arrow like that; we added it to the figure to be helpful ☺]. If you look even more carefully there, you can see that Wireshark is being really helpful to you as well, telling you that the first TCP segment containing the beginning of the POST message is packet #4 in the particular trace for the example in Figure 3, which is the trace *tcp-wireshark-trace1-1* noted in footnote 2. The second TCP segment containing the POST message in packet #5 in the trace, and so on.

Let's now "get our hands dirty" by looking at some TCP segments.

- First, filter the packets displayed in the Wireshark window by entering "tcp" (lowercase, no quotes, and don't forget to press return after entering!) into the display filter specification window towards the top of the Wireshark window. Your Wireshark display should look something like Figure 4. In Figure 4, we've noted the TCP segment that has its SYN bit set – this is the first TCP message in the three-way handshake that sets up the TCP connection to gaia.cs.umass.edu that will eventually carry the HTTP POST message and the alice.txt file. We've also noted the SYNACK segment (the second step in TCP three-way handshake), as well as the TCP segment (packet #4, as discussed above) that carries the POST message and the beginning of the alice.txt file. Of course, if you're taking your own trace file, the packet numbers will be different, but you should see similar behavior to that shown in Figures 3 and 4.

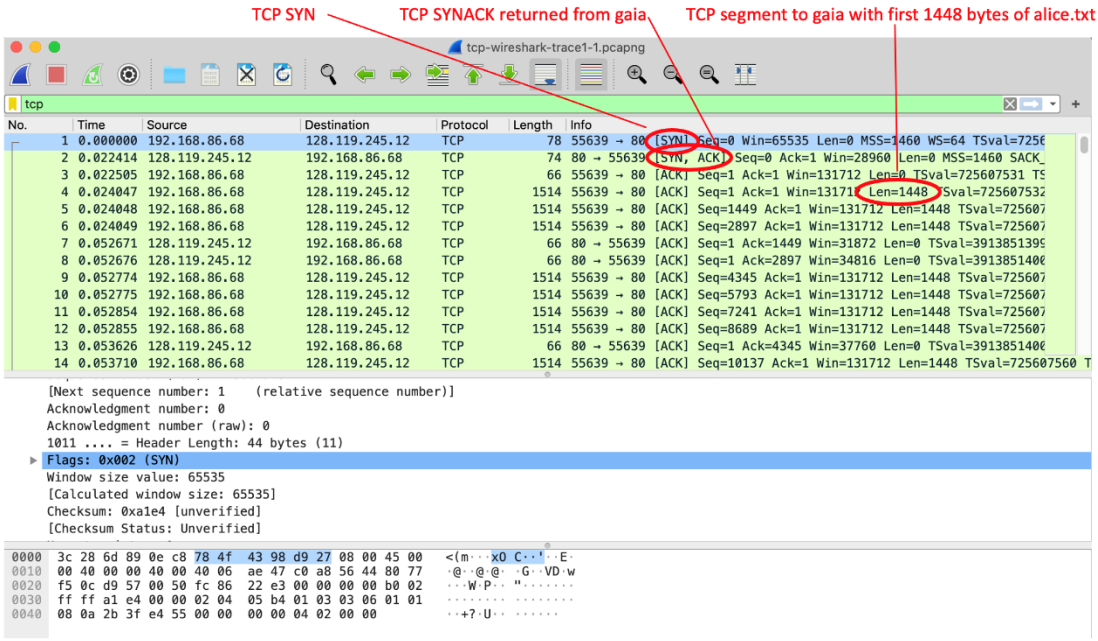


Figure 4: TCP segments involved in sending the HTTP POST message (including the file alice.txt) to gaia.cs.umass.edu

Answer the following questions[7], either from your own live trace, or by opening the Wireshark captured packet file *tcp-wireshark-trace1-1* in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip

1. What is the IP address and TCP port number used by the client computer (source) that is transferring the alice.txt file to gaia.cs.umass.edu? To answer this question, it's probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "details of the selected packet header window" (refer to Figure 2 in the "Getting Started with Wireshark" Lab if you're uncertain about the Wireshark windows).
2. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

[7] For the author's class, when answering the following questions with hand-in assignments, students sometimes need to print out specific packets (see the introductory Wireshark lab for an explanation of how to do this) and indicate where in the packet they've found the information that answers a question. They do this by marking paper copies with a pen or annotating electronic copies with text in a colored font. There are also learning management system (LMS) modules for teachers that allow students to answer these questions online and have answers auto-graded for these Wireshark labs at http://gaia.cs.umass.edu/kurose_ross/lms.htm

UDP

Start capturing packets in Wireshark and then do something that will cause your host to send and receive several UDP packets. It's also likely that just by doing nothing (except capturing packets via Wireshark) that some UDP packets sent by others will appear in your trace. In particular, the Domain Name System (DNS) protocol (see section 2.4 in the text; and the DNS Wireshark Lab) typically sends DNS query and response messages inside of UDP, so it's likely that you'll find some DNS messages (and therefore UDP packets) in your trace.

Specifically you can try out the nslookup command, which invokes the underlying DNS protocol, which in turn will send UDP segments from/to the host issuing the nslookup. nslookup is available in most Microsoft, Apple IOS, and Linux operating systems. To run nslookup you just type the nslookup command on the command line in a DOS window, Mac IOS terminal window, or Linux shell. Figure 1 is a screenshot of running nslookup on the Linux command line on the newworld.cs.umass.edu host located in the CS Department at the University of Massachusetts (UMass) campus, to display the IP address of www.nyu.edu.

```
newworld.cs.umass.edu> nslookup www.nyu.edu
Server:         128.119.240.1
Address:        128.119.240.1#53

Non-authoritative answer:
www.nyu.edu     canonical name = WEB.GSLB.nyu.edu.
Name:   WEB.GSLB.nyu.edu
Address: 216.165.47.12
Name:   WEB.GSLB.nyu.edu
Address: 2607:f600:1002:6113::100
```

Figure 1: the basic nslookup command

We don't need to go into any more details about nslookup or DNS, as we're just interested in getting a few UDP segments into Wireshark, and we promised this lab would be short!

After starting packet capture on Wireshark, run nslookup for a hostname that you haven't visited for a while. Then stop packet capture, set your Wireshark packet filter so that Wireshark only displays the UDP segments sent and received at your host. Pick the first UDP segment and expand the UDP fields in the details window. If you are unable to find UDP segments in your trace or are unable to run Wireshark on a live network connection, you can download a packet trace containing some UDP segments[8].

[8] You can download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip and extract the trace file dns-wireshark-trace1-1. This trace file can be used to answer this Wireshark lab without actually capturing packets on your own. This trace was made using Wireshark running on one of the author's computers, while performing the steps indicated in this Wireshark lab. Once you've downloaded a trace file, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the trace file name.

Answer the following questions[9]. If you're doing this lab as part of class, your teacher will provide details about how to hand in assignments, whether written or in an LMS.

1. Select the first UDP segment in your trace. What is the packet number[10] of this segment in the trace file? What type of application-layer payload or protocol message is being carried in this UDP segment? Look at the details of this packet in Wireshark. How many fields there are in the UDP header? (You shouldn't look in the textbook! Answer these questions directly from what you observe in the packet trace.) What are the names of these fields?
2. By consulting the displayed information in Wireshark's packet content field for this packet (or by consulting the textbook), what is the length (in bytes) of each of the UDP header fields?
3. The value in the Length field is the length of what? (You can consult the text for this answer). Verify your claim with your captured UDP packet.
4. What is the maximum number of bytes that can be included in a UDP payload? (Hint: the answer to this question can be determined by your answer to 2. above)
5. What is the largest possible source port number? (Hint: see the hint in 4.)
6. What is the protocol number for UDP? Give your answer in decimal notation. To answer this question, you'll need to look into the Protocol field of the IP datagram containing this UDP segment (see Figure 4.13 in the text, and the discussion of IP header fields).
7. Examine the pair of UDP packets in which your host sends the first UDP packet and the second UDP packet is a reply to this first UDP packet. (Hint: for a second packet to be sent in response to a first packet, the sender of the first packet should be the destination of the second packet). What is the packet number[11] of the first of these two UDP segments in the trace file? What is the value in the source port field in this UDP segment? What is the value in the destination port field in this UDP segment? What is the packet number[12] of the second of these two UDP segments in the trace file? What is the value in the source port field in this second UDP segment? What is the value in the destination port field in this second UDP segment? Describe the relationship between the port numbers in the two packets.

---

[9] For the author's class, when answering the following questions with hand-in assignments, students sometimes need to print out specific packets (see the introductory Wireshark lab for an explanation of how to do this) and indicate where in the packet they've found the information that answers a question. They do this by marking paper copies with a pen or annotating electronic copies with text in a colored font. There are also learning management system (LMS) modules for teachers that allow students to answer these questions online and have answers auto-graded for these Wireshark labs at http://gaia.cs.umass.edu/kurose_ross/lms.htm

[10] Remember that this "packet number" is assigned by Wireshark for listing purposes only; it is NOT a packet number contained in any real packet header.

[11] Remember that this "packet number" is assigned by Wireshark for listing purposes only; it is NOT a packet number contained in any real packet header.

[12] Remember that this "packet number" is assigned by Wireshark for listing purposes only; it is NOT a packet number contained in any real packet header.

IP

Capturing packets from an execution of traceroute

In order to generate a trace of IPv4 datagrams for the first two parts of this lab, we'll use the traceroute program to send datagrams of two different sizes to gaia.cs.umass.edu. Recall that traceroute operates by first sending one or more datagrams with the time-to-live (TTL) field in the IP header set to 1; it then sends a series of one or more datagrams towards the same destination with a TTL value of 2; it then sends a series of datagrams towards the same destination with a TTL value of 3; and so on. Recall that a router must decrement the TTL in each received datagram by 1 (actually, RFC 791 says that the router must decrement the TTL by *at least* one). If the TTL reaches 0, the router returns an ICMP message (type 11 – TTL-exceeded) to the sending host. As a result of this behavior, a datagram with a TTL of 1 (sent by the host executing traceroute) will cause the router one hop away from the sender to send an ICMP TTL-exceeded message back to the sender; the datagram sent with a TTL of 2 will cause the router two hops away to send an ICMP message back to the sender; the datagram sent with a TTL of 3 will cause the router three hops away to send an ICMP message back to the sender; and so on. In this manner, the host executing traceroute can learn the IP addresses of the routers between itself and the destination by looking at the source IP addresses in the datagrams containing the ICMP TTL-exceeded messages.

Let's run traceroute and have it send datagrams of two different sizes. The larger of the two datagram lengths will require traceroute messages to be fragmented across multiple IPv4 datagrams.

- **Linux/MacOS.** With the Linux/MacOS traceroute command, the size of the UDP datagram sent towards the final destination can be explicitly set by indicating the number of bytes in the datagram; this value is entered in the traceroute command line immediately after the name or address of the destination. For example, to send traceroute datagrams of 2000 bytes towards gaia.cs.umass.edu, the command would be:

%traceroute gaia.cs.umass.edu 2000

- **Windows.** The tracert program provided with Windows does not allow one to change the size of the ICMP message sent by tracert. So it won't be possible to use a Windows machine to generate ICMP messages that are large enough to force IP fragmentation. However, you can use tracert to generate small, fixed length packets to perform Part 1 of this lab. At the DOS command prompt enter:

>tracert gaia.cs.umass.edu

If you want to do the second part of this lab, you can download a packet trace file that was captured on one of the author's computers[13].

Do the following:

- Start up Wireshark and begin packet capture. *(Capture->Start* or click on the blue shark fin button in the top left of the Wireshark window).
- Enter two traceroute commands, using gaia.cs.umass.edu as the destination, the first with a length of 56 bytes. Once that command has finished executing, enter a second traceroute command for the same destination, but with a length of 3000 bytes.
- Stop Wireshark tracing.

If you're unable to run Wireshark on a live network connection, you can use the packet trace file, *ip-wireshark-trace1-1.pcapng*, referenced in footnote 2. You may well find it valuable to download this trace even if you've captured your own trace and use it, as well as your own trace, as you explore the questions below.

Part 1: Basic IPv4

In your trace, you should be able to see the series of UDP segments (in the case of MacOS/Linux) or ICMP Echo Request messages (Windows) sent by traceroute on your computer, and the ICMP TTL-exceeded messages returned to your computer by the intermediate routers. In the questions below, we'll assume you're using a MacOS/Linux computer; the corresponding questions for the case of a Windows machine should be clear. Your screen should look similar to the screenshot in Figure 2, where we have used the display filter "udp||icmp" (see the light-green-filled display-filter field in Figure 2) so that only UDP and/or ICMP protocol packets are displayed.

[13] You can download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip and extract the trace file *ip-wireshark-trace1-1.pcapng*. This trace file can be used to answer these Wireshark lab questions without actually capturing packets on your own. The trace was made using Wireshark running on one of the author's computers, while performing the steps in this Wireshark lab. Once you've downloaded a trace file, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the trace file name.

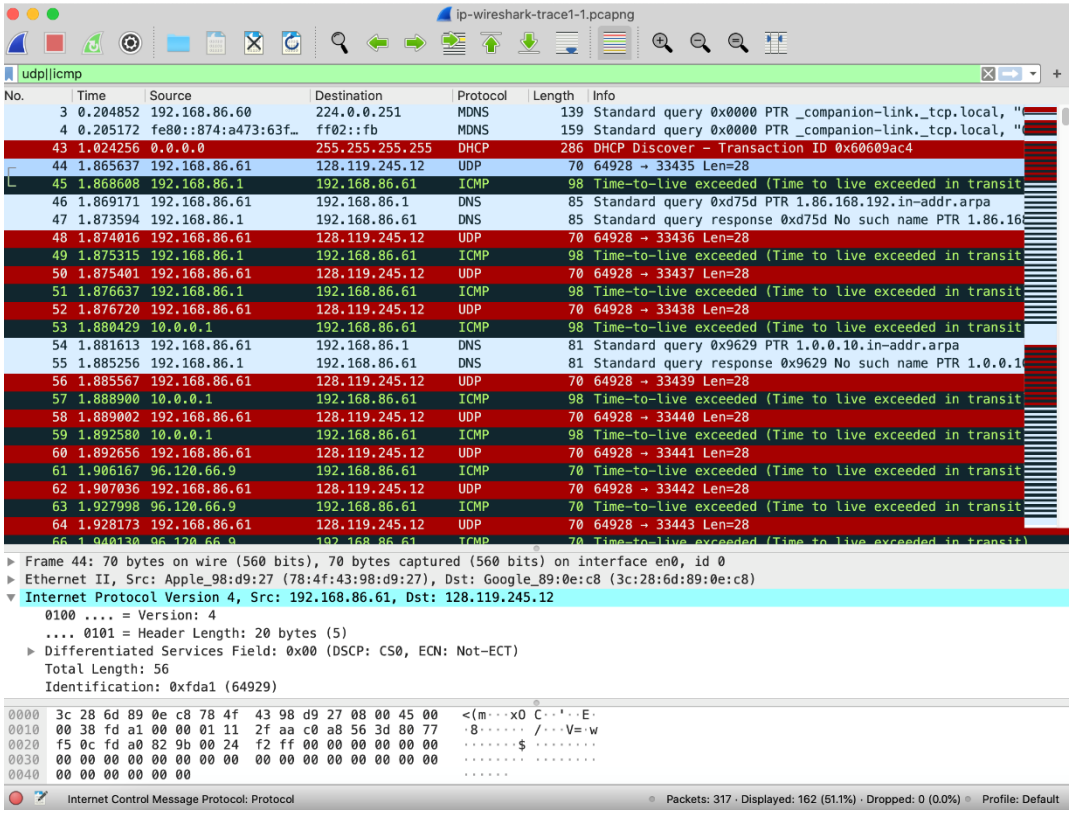


Figure 2: Wireshark screenshot, showing UDP and ICMP packets in the tracefile *ip-wireshark-trace1-1.pcapng*

Answer the following questions[14]. If you're doing this lab as part of class, your teacher will provide details about how to hand in assignments, whether written or in an LMS.

1. Select the first UDP segment sent by your computer via the traceroute command to gaia.cs.umass.edu. (Hint: this is 44th packet in the trace file in the *ip-wireshark-trace1-1.pcapng* file in footnote 2). Expand the Internet Protocol part of the packet in the packet details window. What is the IP address of your computer?
2. What is the value in the time-to-live (TTL) field in this IPv4 datagram's header?
3. What is the value in the upper layer protocol field in this IPv4 datagram's header? [Note: the answers for Linux/MacOS differ from Windows here].
4. How many bytes are in the IP header?
5. How many bytes are in the payload of the IP datagram? Explain how you determined the number of payload bytes.

[14] For the author's class, when answering the following questions with hand-in assignments, students sometimes need to print out specific packets (see the introductory Wireshark lab for an explanation of how to do this) and indicate where in the packet they've found the information that answers a question. They do this by marking paper copies with a pen or annotating electronic copies with text in a colored font. There are also learning management system (LMS) modules for teachers that allow students to answer these questions online and have answers auto-graded for these Wireshark labs at http://gaia.cs.umass.edu/kurose_ross/lms.htm

6. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.

Next, let's look at the *sequence* of UDP segments being sent from your computer via traceroute, destined to 128.119.245.12. The display filter that you can enter to do this is "ip.src==192.168.86.61 and ip.dst==128.119.245.12 and udp and !icmp". This will allow you to easily move sequentially through just the datagrams containing just these segments. Your screen should look similar to Figure 3.

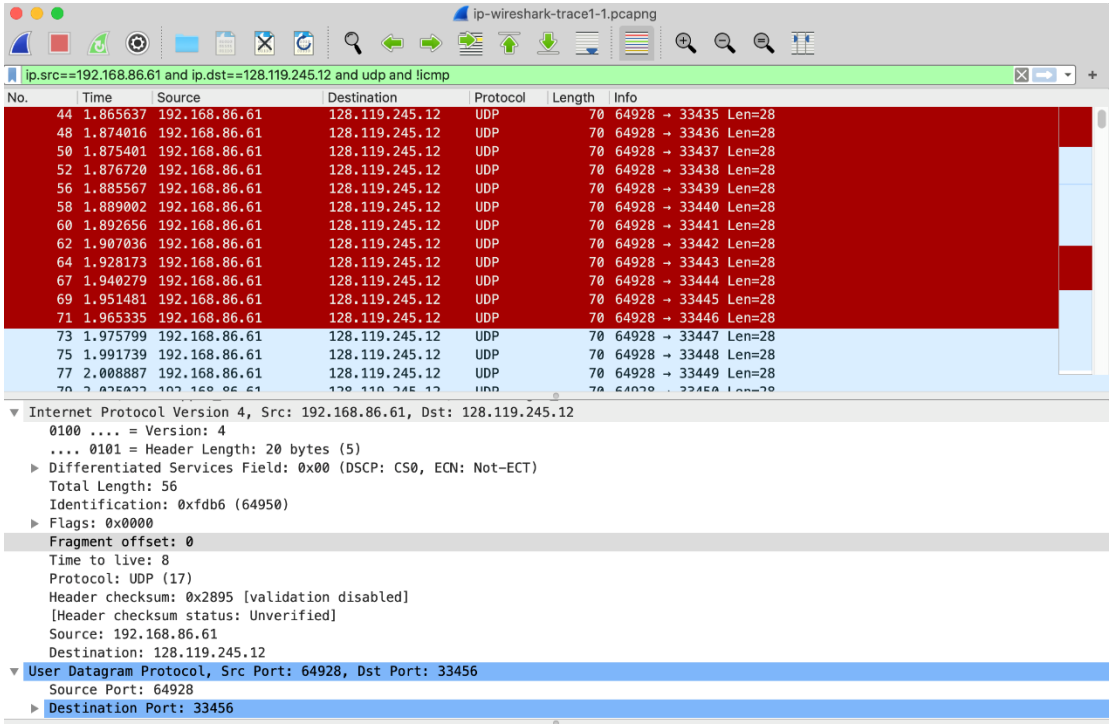


Figure 3: Wireshark screen shot, showing up segments in the tracefile *ip-wireshark-trace1-1.pcapng* using the display filter ip.src==192.168.86.61 and ip.dst==128.119.245.12 and udp and !icmp

7. Which fields in the IP datagram *always* change from one datagram to the next within this series of UDP segments sent by your computer destined to 128.119.245.12, via traceroute? Why?
8. Which fields in this sequence of IP datagrams (containing UDP segments) stay constant? Why?
9. Describe the pattern you see in the values in the Identification field of the IP datagrams being sent by your computer.

Now let's take a look at the ICMP packets being returned to your computer by the intervening routers where the TTL value was decremented to zero (and hence caused the ICMP error message to be returned to your computer). The display filter that you can use to show just these packets is "ip.dst==192.168.86.61 and icmp".

10. What is the upper layer protocol specified in the IP datagrams returned from the routers? [Note: the answers for Linux/MacOS differ from Windows here].
11. Are the values in the Identification fields (across the sequence of all of ICMP packets from all of the routers) similar in behavior to your answer to question 9 above?
12. Are the values of the TTL fields similar, across all of ICMP packets from all of the routers?

Part 2: Fragmentation

In this section, we'll look at a large (3000-byte) UDP segment sent by the traceroute program that is fragmented into multiple IP datagrams. We recommend that you first consult the material on IP fragmentation from the 7th edition of our textbook (and earlier) at http://gaia.cs.umass.edu/kurose_ross/Kurose_Ross_7th_edition_section_4.3.2.pdf

Sort the packet listing from Part 1, with any display filters cleared, according to time, by clicking on the *Time* column.

13. Find the first IP datagram containing the first part of the segment sent to 128.119.245.12 sent by your computer via the traceroute command to gaia.cs.umass.edu, *after* you specified that the traceroute packet length should be 3000. (Hint: This is packet 179 in the *ip-wireshark-trace1-1.pcapng* trace file in footnote 2. Packets 179, 180, and 181 are three IP datagrams created by fragmenting the first single 3000-byte UDP segment sent to 128.119.145.12). Has that segment been fragmented across more than one IP datagram? (Hint: the answer is yes[15]!)
14. What information in the IP header indicates that this datagram been fragmented?
15. What information in the IP header for this packet indicates whether this is the first fragment versus a latter fragment?
16. How many bytes are there in is this IP datagram (header plus payload)?
17. Now inspect the datagram containing the second fragment of the fragmented UDP segment. What information in the IP header indicates that this is *not* the first datagram fragment?
18. What fields change in the IP header between the first and second fragment?
19. Now find the IP datagram containing the third fragment of the original UDP segment. What information in the IP header indicates that this is the last fragment of that segment?

Part 3: IPv6

In this final section we'll take a quick look at the IPv6 datagram using Wireshark. You'll probably want to review section 4.3.4 in the text. Because the Internet is still primarily at IPv4 network (see section 4.3.4), and your own computer or your ISP may not be configured for IPv6, let's look at a trace of already captured packets that contain some IPv6 packets. To generate this trace, our web browser opened the

[15] Note: if you find your packet has not been fragmented, you should download the zip file in footnote 2 and extract the trace file *ip-wireshark-trace1-1.pcapng* . If your computer has an Ethernet or WiFi interface, a packet size of 3000 *should* cause fragmentation.

youtube.com homepage. Youtube (and Google) provide fairly widespread support for IPv6.

Open the file *ip-wireshark-trace2-1.pcapng* in the .zip file of traces given in footnote 2. Your Wireshark display should look similar to Figure 4.

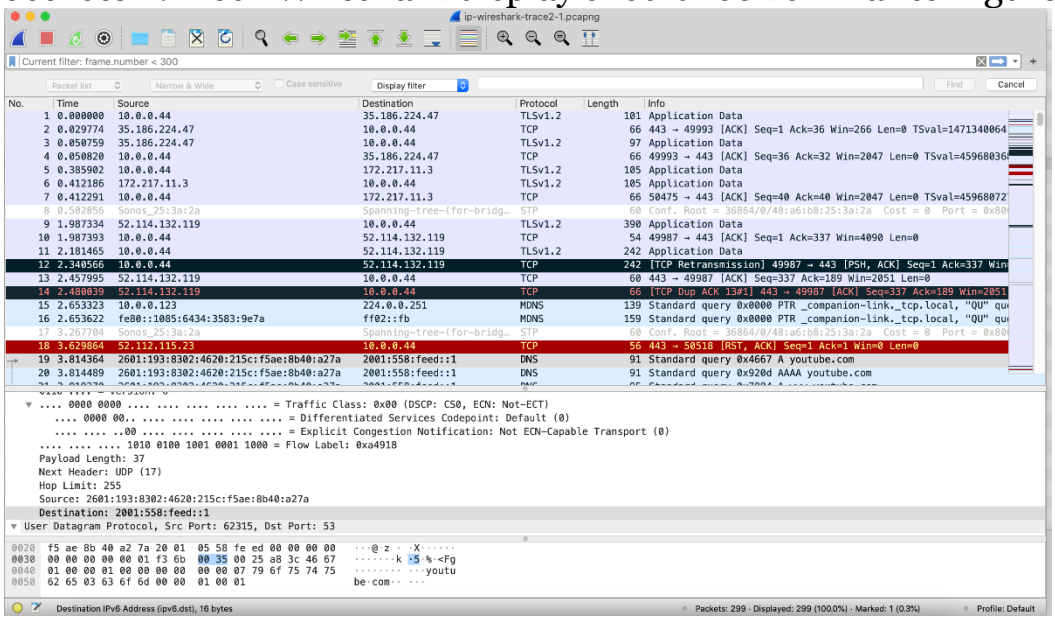


Figure 4: Wireshark screenshot, showing the first set of captured packets in *ip-wireshark-trace2-1.pcapng*. If you look at the Source column, you'll see some IPv6 addresses!

Let's start by taking a closer look at the 20th packet in this trace, sent at t=3.814489. This is a DNS request (contained in an IPv6 datagram) to an IPv6 DNS server for the IPv6 address of youtube.com. The DNS AAAA request type is used to resolve names to IPv6 IP addresses.

Answer the following questions:

20. What is the IPv6 address of the computer making the DNS AAAA request? This is the source address of the 20th packet in the trace. Give the IPv6 source address for this datagram in the exact same form as displayed in the Wireshark window[16].
21. What is the IPv6 destination address for this datagram? Give this IPv6 address in the exact same form as displayed in the Wireshark window.
22. What is the value of the flow label for this datagram?
23. How much payload data is carried in this datagram?
24. What is the upper layer protocol to which this datagram's payload will be delivered at the destination?

[16] Recall that an IPv6 address is shown as 8 sets of 4 hexadecimal digits, with each set separated by colons, and with leading zeros omitted. If an IPv6 address has two colons in a row ('::'), this is shorthand meaning that all of the intervening bytes between the two colons are zero. Thus, for example, `fe80::1085:6434:583:e79` is shorthand for `fe80:0000:0000:0000:1085:6434:0583:0e79`. Make sure you understand this example.

Lastly, find the IPv6 DNS response to the IPv6 DNS AAAA request made in the 20th packet in this trace. This DNS response contains IPv6 addresses for youtube.com.

25. How many IPv6 addresses are returned in the response to this AAAA request?
26. What is the first of the IPv6 addresses returned by the DNS for youtube.com (in the *ip-wireshark-trace2-1.pcapng* trace file, this is also the address that is numerically the smallest)? Give this IPv6 address in the exact same shorthand form as displayed in the Wireshark window.

ICMP

. ICMP and Ping

Let's begin our ICMP adventure by capturing the packets generated by the Ping program. You may recall that the Ping program is simple tool that allows anyone (for example, a network administrator) to verify if a host is live or not. The Ping program in the source host sends a packet to the target IP address; if the target is live, the Ping program in the target host responds by sending a packet back to the source host. As you might have guessed (given that this lab is about ICMP), both of these Ping packets are ICMP packets.

Do the following[17]:

- Let's begin this adventure by opening the Windows Command Prompt application (which can be found in your Accessories folder).
- Start up the Wireshark packet sniffer, and begin Wireshark packet capture.
- The *ping* command is in c:\windows\system32, so type either "*ping –n 10 hostname*" or "*c:\windows\system32\ping –n 10 hostname*" in the MS-DOS command line (without quotation marks), where hostname is a host on another continent. If you're outside of Asia, you may want to enter www.ust.hk for the Web server at Hong Kong University of Science and Technology. The argument "*-n 10*" indicates that 10 ping messages should be sent. Then run the Ping program by typing return.
- When the Ping program terminates, stop the packet capture in Wireshark.

At the end of the experiment, your Command Prompt Window should look something like Figure 1. In this example, the source ping program is in

[17] If you are unable to run Wireshark live on a computer, you can download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and extract the file *ICMP-ethereal-trace-1*. The traces in this zip file were collected by Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you have downloaded the trace, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the *ICMP-ethereal-trace-1* trace file. You can then use this trace file to answer the questions below.

Massachusetts and the destination Ping program is in Hong Kong. From this window we see that the source ping program sent 10 query packets and received 10 responses. Note also that for each response, the source calculates the round-trip time (RTT), which for the 10 packets is on average 375 msec.

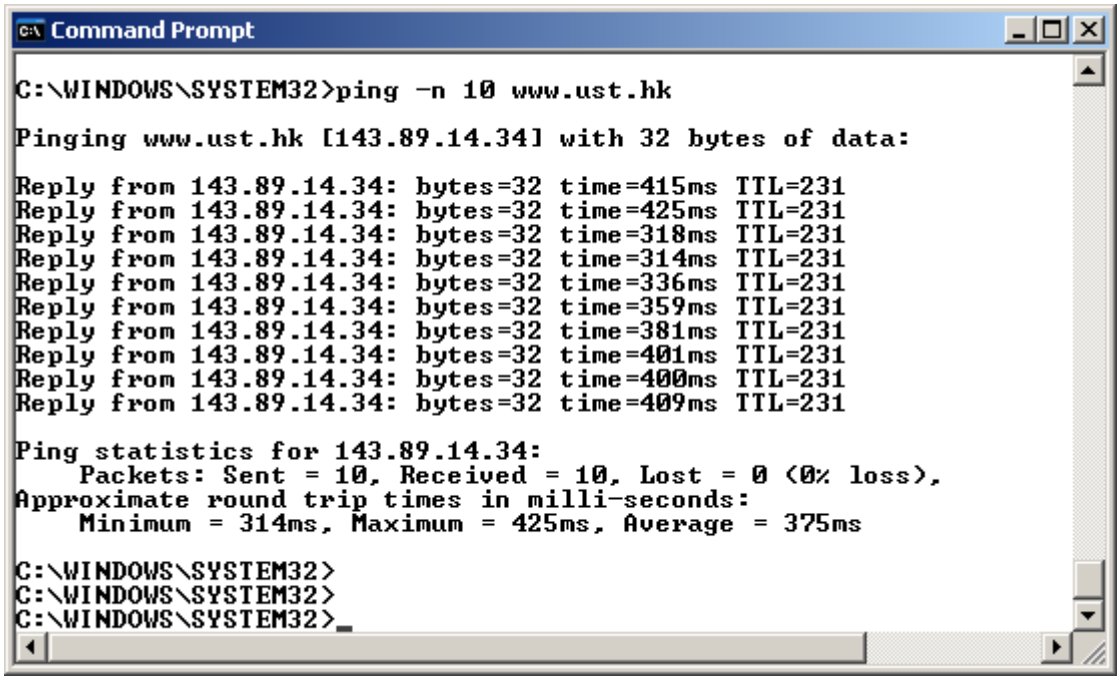


Figure 1 Command Prompt window after entering Ping command.

Figure 2 provides a screenshot of the Wireshark output, after "icmp" has been entered into the filter display window. Note that the packet listing shows 20 packets: the 10 Ping queries sent by the source and the 10 Ping responses received by the source. Also note that the source's IP address is a private address (behind a NAT) of the form 192.168/12; the destination's IP address is that of the Web server at HKUST. Now let's zoom in on the first packet (sent by the client); in the figure below, the packet contents area provides information about this packet. We see that the IP datagram within this packet has protocol number 01, which is the protocol number for ICMP. This means that the payload of the IP datagram is an ICMP packet.

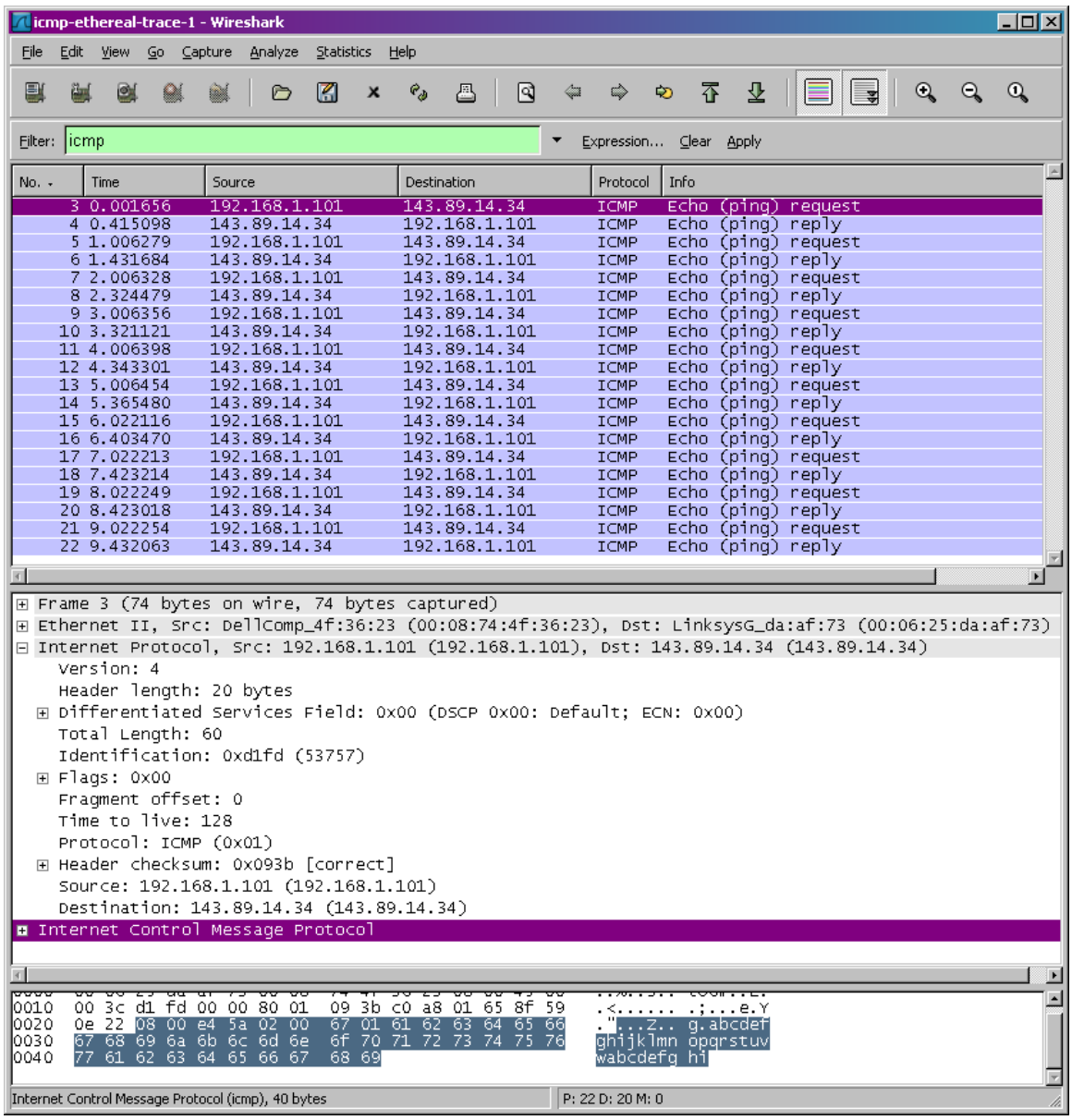


Figure 2 Wireshark output for Ping program with Internet Protocol expanded.

Figure 3 focuses on the same ICMP but has expanded the ICMP protocol information in the packet contents window. Observe that this ICMP packet is of Type 8 and Code 0 - a so-called ICMP “echo request” packet. (See Figure 5.19 of text.) Also note that this ICMP packet contains a checksum, an identifier, and a sequence number.

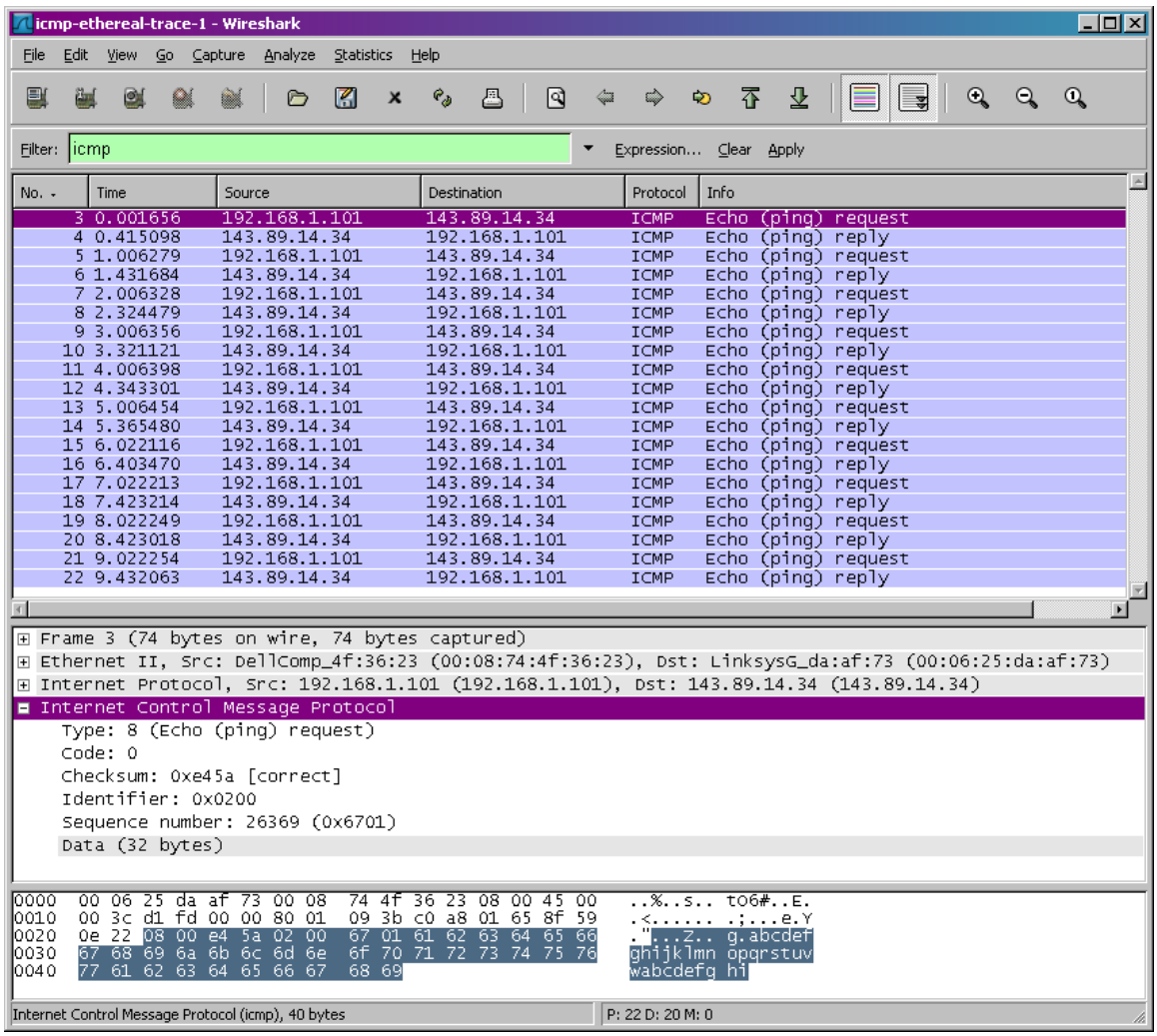


Figure 3 Wireshark capture of ping packet with ICMP packet expanded.

ARP

Let's begin by capturing a set of Ethernet frames to study. To do this, of course, you'll need access to a wired Ethernet connection for your PC or Mac – not necessarily a common scenario these days, given the popularity of wireless WiFi and cellular access. If you're unable to run Wireshark on a live Ethernet connection, you can download a packet trace that was captured while following the steps below on one of the author's computers[18]. In addition, you may well find it valuable to download this trace even if you've captured your own trace and use it, as well as your own trace, when you explore the questions below.

Do the following:

[18] You can download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip and extract the trace file *ethernet-wireshark-trace1*. This trace file can be used to answer this Wireshark lab without actually capturing packets on your own. This trace was made using Wireshark running on one of the author's computers, while performing the steps indicated in this Wireshark lab. Once you've downloaded a trace file, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the trace file name.

- First, make sure your browser's cache of previously downloaded documents is empty.
- Start up Wireshark and enter the following URL into your browser: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html. Your browser should display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture.

First, find the packet number (the leftmost column in the upper Wireshark window) of the HTTP GET message that was sent from your computer to gaia.cs.umass.edu, as well as the beginning of the HTTP response message sent to your computer by gaia.cs.umass.edu. You should see a screen that looks something like this (where packet 126 in the screen shot below contains the HTTP GET message)

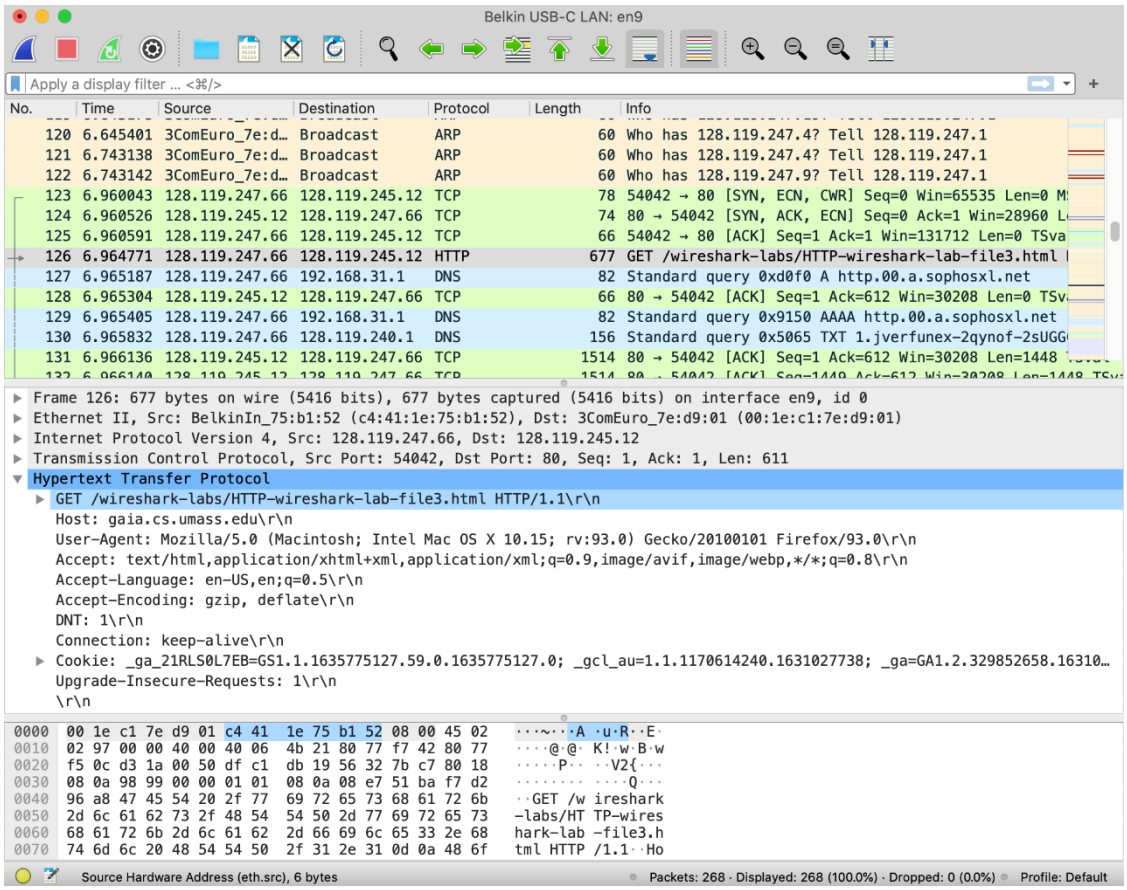


Figure 1: Wireshark display showing HTTP GET message for http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html

Since this lab is about Ethernet and ARP, we're not interested in high-level protocols like IP, TCP or HTTP. We're interested in Ethernet frames and ARP messages!

Let's start by looking at the Ethernet frame containing the HTTP GET message. (Recall that the HTTP GET message is carried inside of a TCP segment, which is carried inside of an IP datagram, which is carried inside of an Ethernet frame;

reread section 1.5.2 in the text if you find this notion of encapsulation a bit confusing). Expand the Ethernet II information in the packet details window. Note that the contents of the Ethernet frame (header as well as payload) are displayed in the packet contents window. Your display should look similar to that shown in Figure 2.

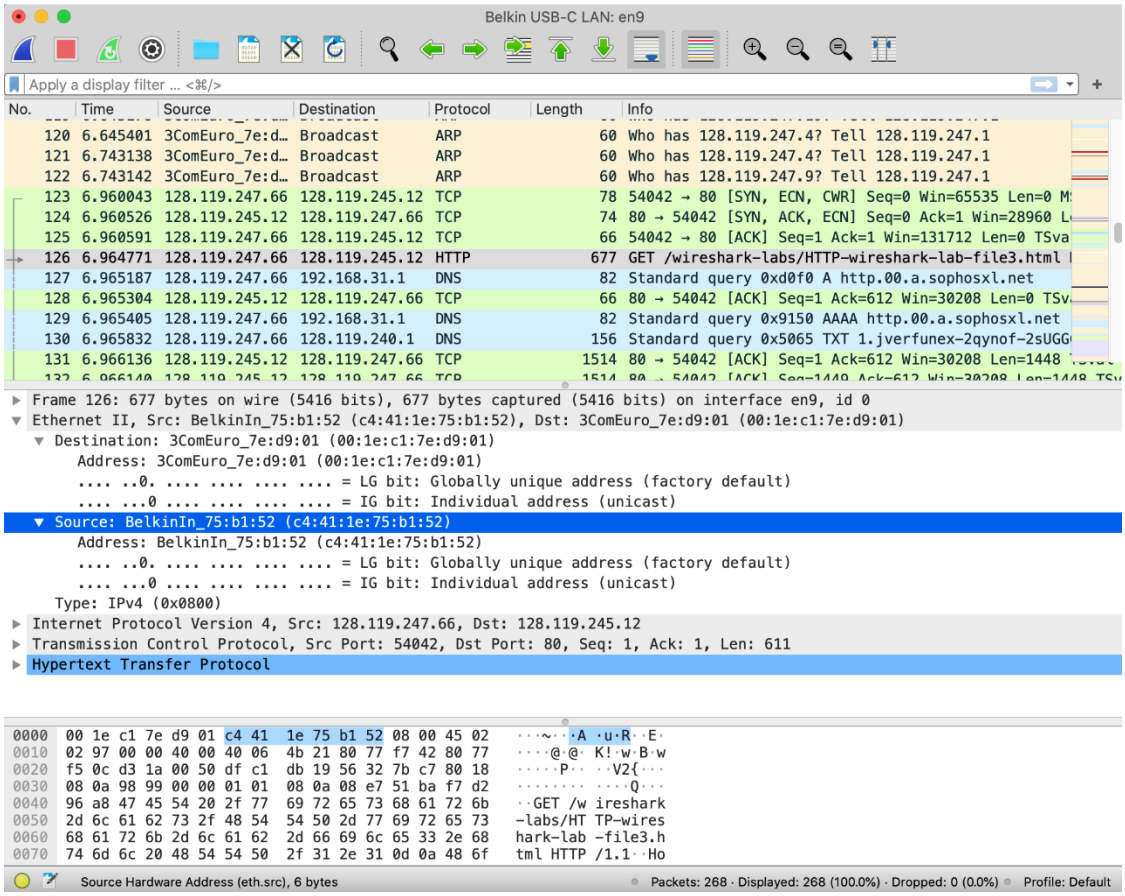


Figure 2: Wireshark display showing details of the Ethernet frame containing the HTTP GET request.

In answering the questions below[19], you can use either your own live trace, or use the Wireshark captured packet file *Ethernet-wireshark-trace1* in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip

- What is the 48-bit Ethernet address of your computer?
- What is the 48-bit destination address in the Ethernet frame? Is this the Ethernet address of gaia.cs.umass.edu? (Hint: the answer is *no*). What device has this as its Ethernet address? [Note: this is an important question, and one that students sometimes get wrong. Re-read pages 483-484 in the text and make sure you understand the answer here.]

[19] For the author's class, when answering the following questions with hand-in assignments, students sometimes need to print out specific packets (see the introductory Wireshark lab for an explanation of how to do this) and indicate where in the packet they've found the information that answers a question. They do this by marking paper copies with a pen or annotating electronic copies with text in a colored font. There are also learning management system (LMS) modules for teachers that allow students to answer these questions online and have answers auto-graded for these Wireshark labs at http://gaia.cs.umass.edu/kurose_ross/lms.htm

- What is the hexadecimal value for the two-byte Frame type field in the Ethernet frame carrying the HTTP GET request? What upper layer protocol does this correspond to?
- How many bytes from the very start of the Ethernet frame does the ASCII "G" in "GET" appear in the Ethernet frame? Do not count any preamble bits in your count, i.e., assume that the Ethernet frame begins with the Ethernet frame's destination address.

Next, answer the following questions, based on the contents of the Ethernet frame containing the first byte of the HTTP response message.

- What is the value of the Ethernet source address? Is this the address of your computer, or of gaia.cs.umass.edu (Hint: the answer is *no*). What device has this as its Ethernet address?
- What is the destination address in the Ethernet frame? Is this the Ethernet address of your computer?
- Give the hexadecimal value for the two-byte Frame type field. What upper layer protocol does this correspond to?
- How many bytes from the very start of the Ethernet frame does the ASCII "O" in "OK" (i.e., the HTTP response code) appear in the Ethernet frame? Do not count any preamble bits in your count, i.e., assume that the Ethernet frame begins with the Ethernet frame's destination address.
- How many Ethernet frames (each containing an IP datagram, each containing a TCP segment) carry data that is part of the complete HTTP "OK 200 ..." reply message?

2. The Address Resolution Protocol

In this section, we'll observe the ARP protocol in action. We strongly recommend that you re-read section 6.4.1 in the text before proceeding.

ARP Caching

Recall that the ARP protocol typically maintains a cache of IP-to-Ethernet address translation pairs on your computer. The *arp* command (in both DOS, MacOS and Linux) is used to view and manipulate the contents of this cache. Since the *arp* command and the ARP protocol have the same name, it's understandably easy to confuse them. But keep in mind that they are different - the *arp* command is used to view and manipulate the ARP cache contents, while the ARP protocol defines the format and meaning of the messages sent and received, and defines the actions taken on ARP message transmission and receipt.

Let's take a look at the contents of the ARP cache on your computer. In DOS, MacOS, and Linux, the "arp -a" command will display the contents of the ARP

cache on your computer. So at a command line, type "arp -a". The results of entering this command on one of the authors' computers is shown in Figure 3.

```
kurose@noho4 ~ %
kurose@noho4 ~ %
kurose@noho4 ~ % arp -a
gw-vlan-2471.cs.umass.edu (128.119.247.1) at 0:1e:c1:7e:d9:1 on en9 ifscope [ethernet]
sammac.cs.umass.edu (128.119.247.19) at (incomplete) on en9 ifscope [ethernet]
robomac.cs.umass.edu (128.119.247.79) at 78:7b:8a:ac:ad:e1 on en9 ifscope [ethernet]
kurose@noho4 ~ %
kurose@noho4 ~ %
kurose@noho4 ~ %
kurose@noho4 ~ %
```

Figure 3: Executing the "arp -a" command from one of the authors' computers

- How many entries are stored in your ARP cache?
- What is contained in each displayed entry of the ARP cache?

In order to observe your computer sending and receiving ARP messages, we'll need to clear the ARP cache, since otherwise your computer is likely to find a needed IP-Ethernet address translation pair in its cache and consequently not need to send out an ARP message. The "*arp –d -a*" command will clear your ARP cache using the command line. In order to run this command on a Mac or Linux machine you'll need root privileges or use *sudo*. If you don't have root privileges and can't run Wireshark on a Windows machine, you can skip the trace collection part of this lab and use the trace file *ethernet-wireshark-trace1* discussed earlier.

Observing ARP in action

Do the following[20]:

- Clear your ARP cache, as described above and make sure your browser's cache is cleared of previously downloaded documents.
- Start up the Wireshark packet sniffer.
- Enter the following URL into your browser:
  http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-lab-file3.html
  Your browser should again display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture.

Again, we're not interested in IP or higher-layer protocols, so let's just look at ARP packets. Your display should look similar to that shown in Figure 3 (note we have entered "arp" into the display filter window at the top of the Wireshark screen).

[20] The *ethernet-wireshark-trace-1* trace file in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip and was created using the steps below (in particular after the ARP cache had been flushed).

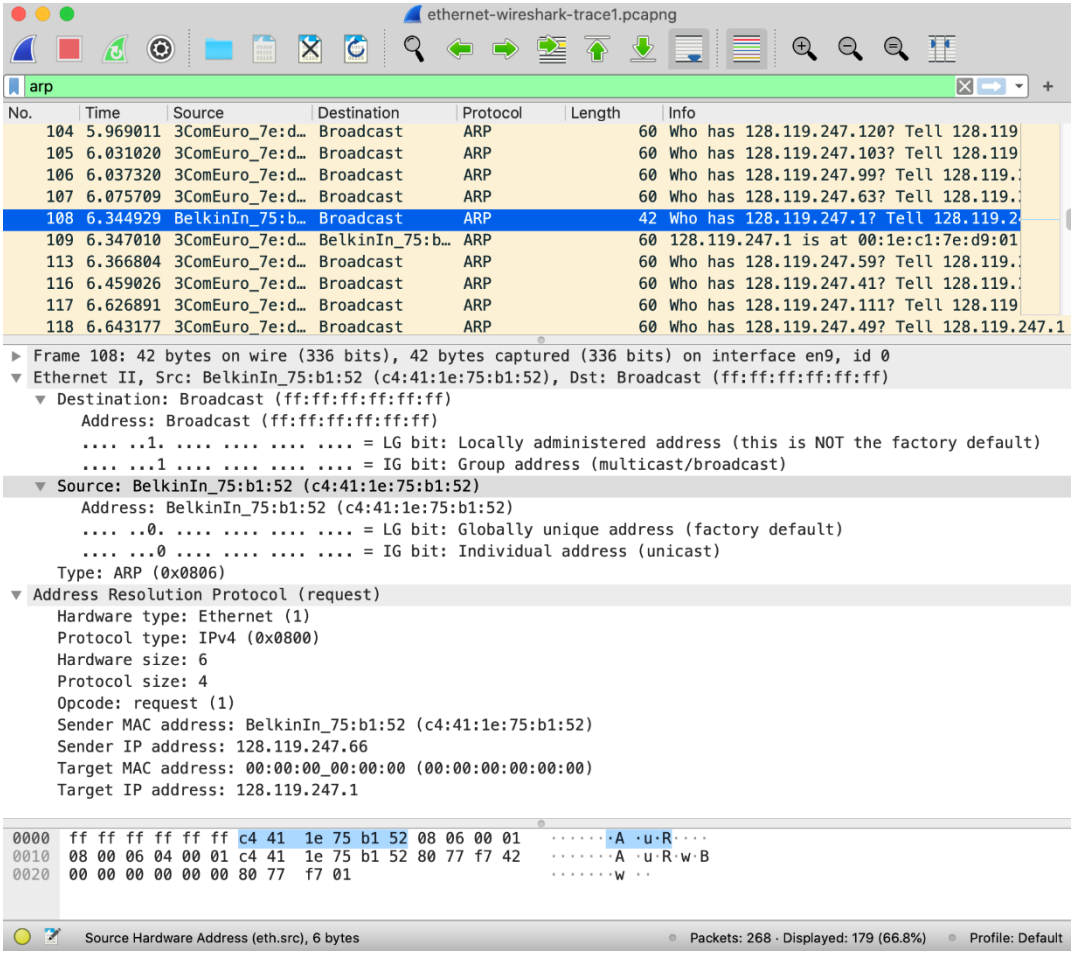


Figure 3: An ARP query being broadcast from the computer of one of the authors'

Let's start by looking at the Ethernet frames containing ARP messages. Answer the following questions:

- What is the hexadecimal value of the source address in the Ethernet frame containing the ARP request message sent out by your computer?
- What is the hexadecimal value of the destination addresses in the Ethernet frame containing the ARP request message sent out by your computer? And what device(if any) corresponds to that address (e.g,, client, server, router, switch or otherwise…)?
- What is the hexadecimal value for the two-byte Ethernet Frame *type* field. What upper layer protocol does this correspond to?

Now let's dig even a bit deeper into the ARP messages themselves. To answer this question, you will need to dig into ARP. The original RFC (https://datatracker.ietf.org/doc/html/rfc826) that defines ARP is a little hard to read. The Wikipedia entry for ARP is pretty good: https://en.wikipedia.org/wiki/Address_Resolution_Protocol
Answer the following question about the ARP request message sent by your computer.

- How many bytes from the very beginning of the Ethernet frame does the ARP *opcode* field begin?

- What is the value of the *opcode* field within the ARP request message sent by your computer?
- Does the ARP request message contain the IP address of the sender? If the answer is yes, what is that value?
- What is the IP address of the device whose corresponding Ethernet address is being requested in the ARP request message sent by your computer?

Now find the ARP reply message that was sent in response to the ARP request from your computer.

- What is the value of the *opcode* field within the ARP reply message received by your computer?
- *Finally (!),* let's look at the **answer** to the ARP request message! What is the Ethernet address corresponding to the IP address that was specified in the ARP request message sent by your computer (see question 18)?

We've looked the ARP request message sent by your computer running Wireshark, and the ARP reply sent in reply. But there are other devices in this network that are also sending ARP requests that you can find in the trace.

- We've looked the ARP request message sent by your computer running Wireshark, and the ARP reply message sent in response. But there are other devices in this network that are also sending ARP request messages that you can find in the trace. Why are there no ARP replies in your trace that are sent in response to these other ARP request messages?

DHCP

1. In a terminal window/shell, enter the following commands:
   ```
   sudo ip addr flush en0
   sudo dhclient -r
   ```
   where `en0` (in this example) is the interface on which you want to capture packets using Wireshark. You can easily find the list of interface names in Wireshark by choosing Capture -> Options. This command will remove the existing IP address of the interface, and release any existing DHCP address leases.
2. Start up Wireshark, capturing packets in the interface you de-configured in Step 1.
3. In the terminal window/shell, enter the following command:
   ```
   sudo dhclient en0
   ```
   where, as with above, `en0` is the interface on which you are currently capturing packets. This will cause the DHCP protocol to request and receive an IP address and other information from the DHCP server.
4. After waiting for a few seconds, stop Wireshark capture.

On a PC:

1. In a command-line window enter the following command:
   ```
   > ipconfig /release
   ```
   This command will cause your PC to give up its IP address.
2. Start up Wireshark.
3. In the command-line window enter the following command:
   ```
   > ipconfig /renew
   ```
   This will cause the DHCP protocol to request and receive an IP address and other information from a DHCP server.
4. After waiting for a few seconds, stop Wireshark capture.

After stopping Wireshark capture in step 4, you should take a peek in your Wireshark window to make sure you've actually captured the packets that we're looking for. If you enter "dhcp" into the display filter field (as shown in the light green field in the top left of Figure 1), your screen (on a Mac) should look similar to Figure 1.

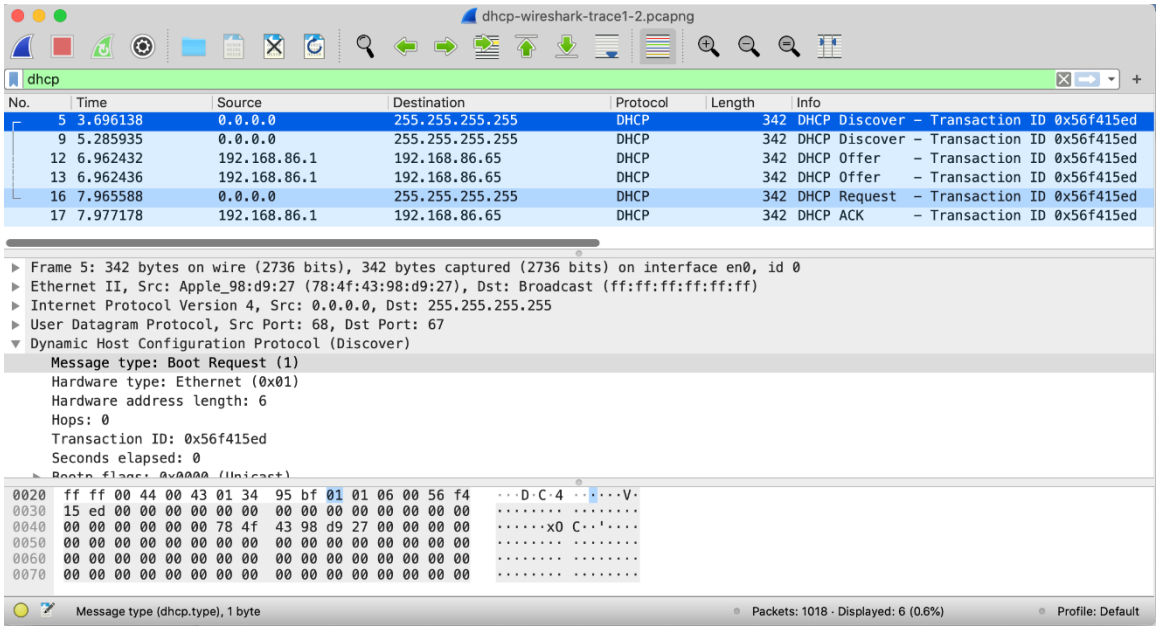


Figure 1: Wireshark display, showing the capture of DHCP Discover, Offer, Request and ACK messages

If you're unable to run Wireshark on a live network connection, are unable to capture all four DHCP messages, or are assigned to do so by your instructor, you can use the Wireshark trace file, *dhcp-wireshark-trace1-1.pcapng*[21] that we've gathered following the steps above on one of the author's computers. You may well find it valuable to download this trace even if you've captured your own trace and use it, as well as your own trace, as you explore the questions below.

DHCP Questions

Answer the following questions[22]. If you're doing this lab as part of class, your teacher will provide details about how to hand in assignments, whether written or in an LMS.

Let's start by looking at the DHCP Discover message. Locate the IP datagram containing the first Discover message in your trace.

1. Is this DHCP Discover message sent out using UDP or TCP as the underlying transport protocol?
2. What is the source IP address used in the IP datagram containing the Discover message? Is there anything special about this address? Explain.
3. What is the destination IP address used in the datagram containing the Discover message. Is there anything special about this address? Explain.
4. What is the value in the transaction ID field of this DHCP Discover message?
5. Now inspect the options field in the DHCP Discover message. What are five pieces of information (beyond an IP address) that the client is suggesting or requesting to receive from the DHCP server as part of this DHCP transaction?

Now let's look at the DHCP Offer message. Locate the IP datagram containing the DHCP Offer message in your trace that was sent by a DHCP server in the response to the DHCP Discover message that you studied in questions 1-5 above.

6. How do you know that this Offer message is being sent in response to the DHCP Discover message you studied in questions 1-5 above?
7. What is the *source* IP address used in the IP datagram containing the Offer message? Is there anything special about this address? Explain.
8. What is the *destination* IP address used in the datagram containing the Offer message? Is there anything special about this address? Explain. [Hint: Look at your trace carefully. The answer to this question may differ from what you see in Figure 4.24 in the textbook. If you really want to dig into this, consult the DHCP RFC, page 24.]

---

[21] You can download the zip file http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces-8.1.zip and extract the trace file *dhcp-wireshark-trace1-1.pcapng*. These trace files can be used to answer these Wireshark lab questions without actually capturing packets on your own. Each trace was made using Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you've downloaded a trace file, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the trace file name.

[22] For the author's class, when answering the following questions with hand-in assignments, students sometimes need to print out specific packets (see the introductory Wireshark lab for an explanation of how to do this) and indicate where in the packet they've found the information that answers a question. They do this by marking paper copies with a pen or annotating electronic copies with text in a colored font. There are also learning management system (LMS) modules for teachers that allow students to answer these questions online and have answers auto-graded for these Wireshark labs at http://gaia.cs.umass.edu/kurose_ross/lms.htm

9. Now inspect the options field in the DHCP Offer message. What are five pieces of information that the DHCP server is providing to the DHCP client in the DHCP Offer message?

It would appear that once the DHCP Offer message is received, that the client may have all of the information it needs to proceed. However, the client may have received OFFERs from multiple DHCP servers and so a second phase is needed, with two more mandatory messages – the client-to-server DHCP Request message, and the server-to-client DHCP ACK message is needed. But at least the client knows there is at least one DHCP server out there! Let's take a look at the DHCP Request message, remembering that although we've already seen a Discover message in our trace, that is not always the case when a DHCP request message is sent.

Locate the IP datagram containing the first DHCP Request message in your trace, and answer the following questions.

10. What is the UDP source port number in the IP datagram containing the first DHCP Request message in your trace? What is the UDP destination port number being used?
11. What is the source IP address in the IP datagram containing this Request message? Is there anything special about this address? Explain.
12. What is the destination IP address used in the datagram containing this Request message. Is there anything special about this address? Explain.
13. What is the value in the transaction ID field of this DHCP Request message? Does it match the transaction IDs of the earlier Discover and Offer messages?
14. Now inspect the options field in the DHCP Discover message and take a close look at the "Parameter Request List". The DHCP RFC notes that

    > "The client can inform the server which configuration parameters the client is interested in by including the 'parameter request list' option. The data portion of this option explicitly lists the options requested by tag number."

    What differences do you see between the entries in the 'parameter request list' option in this Request message and the same list option in the earlier Discover message?

Locate the IP datagram containing the first DHCP ACK message in your trace, and answer the following questions.

15. What is the source IP address in the IP datagram containing this ACK message? Is there anything special about this address? Explain.
16. What is the destination IP address used in the datagram containing this ACK message. Is there anything special about this address? Explain.
17. What is the name of the field in the DHCP ACK message (as indicated in the Wireshark window) that contains the assigned client IP address?
18. For how long a time (the so-called "lease time") has the DHPC server assigned this IP address to the client?
19. What is the IP address (returned by the DHCP server to the DHCP client in this DHCP ACK message) of the first-hop router on the default path from the client to the rest of the Internet?

Reference:

https://gaia.cs.umass.edu/kurose_ross/wireshark.php