Roll No.**24** Exam Seat No._____

VIVEKANANDEDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Hashu Advani Memorial Complex, Collector's Colony, R. C. Marg,Chembur, Mumbai – 400074. Contact No. 02261532532

Since 1962

CERTIFICATE

Certified that Mr. **NARENDER KESWANI [ROLL NO: 24]** of **FYMCA-1B** has satisfactorily completed a course of the necessary experiments in **MCAL26 – NETWORKING WITH LINUX LAB** under the supervision of **Mrs.VAISHALI GATTY** in the Institute of Technology in the academic year **2021- 2022.**

Principal Head of Department

Lab In-charge Subject Teacher

V.E.S. Institute of Technology, Collector Colony, Chembur, Mumbai

Department of M.C.A

NWL INDEX

Sr. No	Contents	Date Of Preparation	Date Of Submission	Marks	Sign
1	Installation of NS-3 in Linux, NetAnim,WireShark	04/05/2022	11/05/2022	10	
2	Program to simulate traffic between two nodes(Point to Point)	11/05/2022	18/05/2022	10	
3	Program to simulate star topology	11/05/2022	18/05/2022	10	
4	Program to simulate bus topology	11/05/2022	18/05/2022	10	
5	Program to simulate hybrid topology(Wireless Network Topology)	18/05/2022	25/05/2022	10	
6	Program to simulate UDP server client	25/05/2022	01/06/2022	10	
7	Program to simulate DHCP server and n clients	01/06/2022	15/06/2022	10	
8	Simulate a simple network in Network Simulator	15/06/2022	22/06/2022	10	
9	Program to simulate FTP using TCP protocol	22/06/2022	29/06/2022	10	
10	Analyze the network traffic and performance parameters of network using WireShark	29/06/2022	05/07/2022	10	
11	Mini project	29/06/2022	06/07/2022		

Aim: Installation of NS-3, NetAnim and Wireshark in Linux

THEORY:

Network Simulator:

Network simulator is a tool used for simulating the real-world network on one computer by writing scripts in C++ or Python. Normally if we want to perform experiments, to see how our network works using various parameters. We don't have required number of computers and routers for making different topologies. Even if we have these resources, it is very expensive to build such a network for experiment purposes.

So, to overcome these drawbacks we used NS3, which is a discrete event network simulator for Internet.

Ns-3:

The ns-3 simulator is a discrete-event network simulator targeted primarily for research and educational use. The ns-3 project, started in 2006, is an open-source project developing ns-3.

NS3 helps to create various virtual nodes (i.e., computers in real life) and with the help of various Helper classes it allows us to install devices, internet stacks, application, etc to our nodes.

Using NS3 we can create PointToPoint, Wireless, CSMA, etc connections between nodes. PointToPoint connection is same as a LAN connected between two computers. Wireless connection is same as WiFi connection between various computers and routers. CSMA connection is same as bus topology between computers. After building connections we try to install NIC to every node to enable network connectivity.

Ns3 gives us special features which can be used for real life integrations.

Features of Ns-3:

1. **Tracing of the nodes:-** NS3 allows us to trace the routes of the nodes which helps us to know how much data is send or received. Trace files are generated to monitor these activities.
2. **NetAnim:-** It stands for Network Animator.It is an animated version of how network will look in real and how data will be transferred from one node to other.
3. **Pcap file:-** NS3 helps to generate pcap file which can be used to get all information of the packets (e.g., Sequence number, Source IP, destination IP, etc). These pcaps can be seen using a software tool known as wireshark.
4. **gnuPlot:-** GnuPlot is used to plot graphs from the data which we get from trace file of NS3. Gnuplot gives more accurate graph compare to other graph making tools and also it is less complex than other tools.

NetAnim:

NetAnim is a stand-alone program which uses the custom trace files generated by the animation interface to graphically display the simulation. NetAnim is based on the multiplatform Qt4 GUI toolkit.

The NetAnim GUI provides play, pause, and record buttons. Play and pause start and stop the simulation. The record button starts a series of screenshots of the animator, which are written to the directory in which the trace file was run. Two slider bars also exist. The top slider provides a "seek" functionality, which allows a user to skip to any moment in the simulation. The bottom slider changes the granularity of the time step for the animation.
Finally, there is a quit button to stop the simulation and quit the animator.

WireShark:

Wireshark is an open-source tool for profiling network traffic and analyzing packets. Such a tool is often referred to as a network analyzer, network protocol analyzer or sniffer.

Uses of Wireshark:

1. It is used by network security engineers to examine security problems.
2. It allows the users to watch all the traffic being passed over the network.
3. It is used by network engineers to troubleshoot network issues.
4. It also helps to troubleshoot latency issues and malicious activities on your network.
5. It can also analyze dropped packets.
6. It helps us to know how all the devices like laptop, mobile phones, desktop, switch, routers, etc., communicate in a local network or the rest of the world.

Functionality of Wireshark:

1. Wireshark is similar to tcpdump in networking. Tcpdump is a common packet analyzer which allows the user to display other packets and TCP/IP packets, being transmitted and received over a network attached to the computer. It has a graphic end and some sorting and filtering functions. Wireshark users can see all the traffic passing through the network.
2. Wireshark can also monitor the unicast traffic which is not sent to the network's MAC address interface. But the switch does not pass all the traffic to the port. Hence, the promiscuous mode is not sufficient to see all the traffic. The various network taps or port mirroring is used to extend capture at any point.
3. Port mirroring is a method to monitor network traffic. When it is enabled, the switch sends the copies of all the network packets present at one port to another port.

Features of Wireshark:

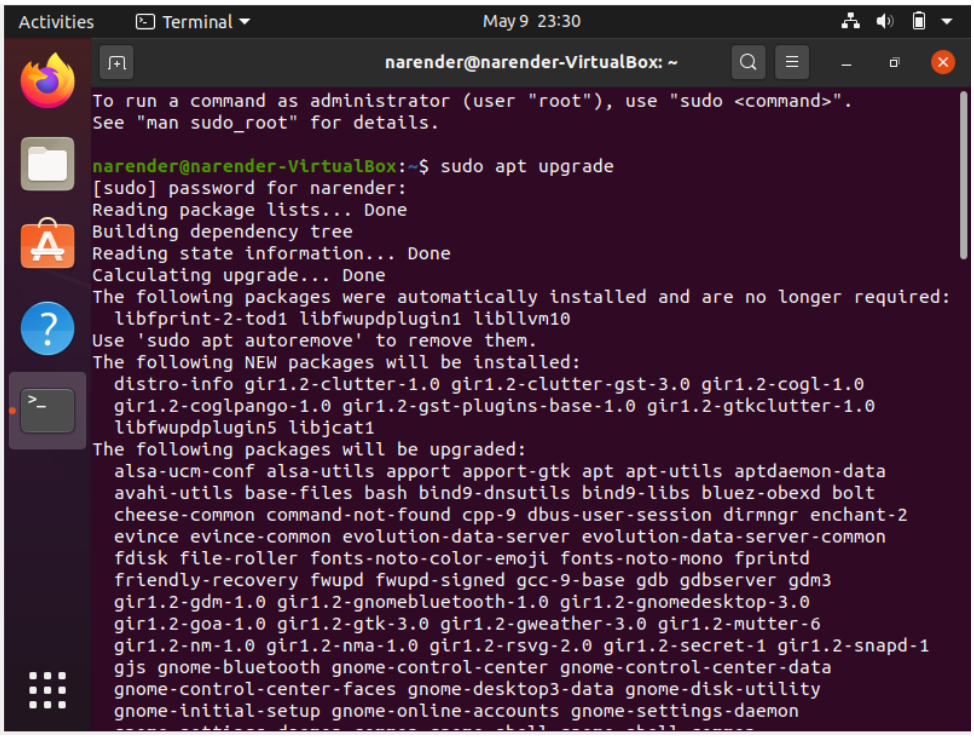
1. It is multi-platform software, i.e., it can run on Linux, Windows, OS X, FreeBSD, NetBSD, etc.
2. It is a standard three-pane packet browser.
3. It performs deep inspection of the hundreds of protocols.
4. It often involves live analysis, i.e., from the different types of the network like the Ethernet, loopback, etc., we can read live data.
5. It has sort and filter options which makes ease to the user to view the data.
6. It is also useful in VoIP analysis.

7. It can also capture raw USB traffic.

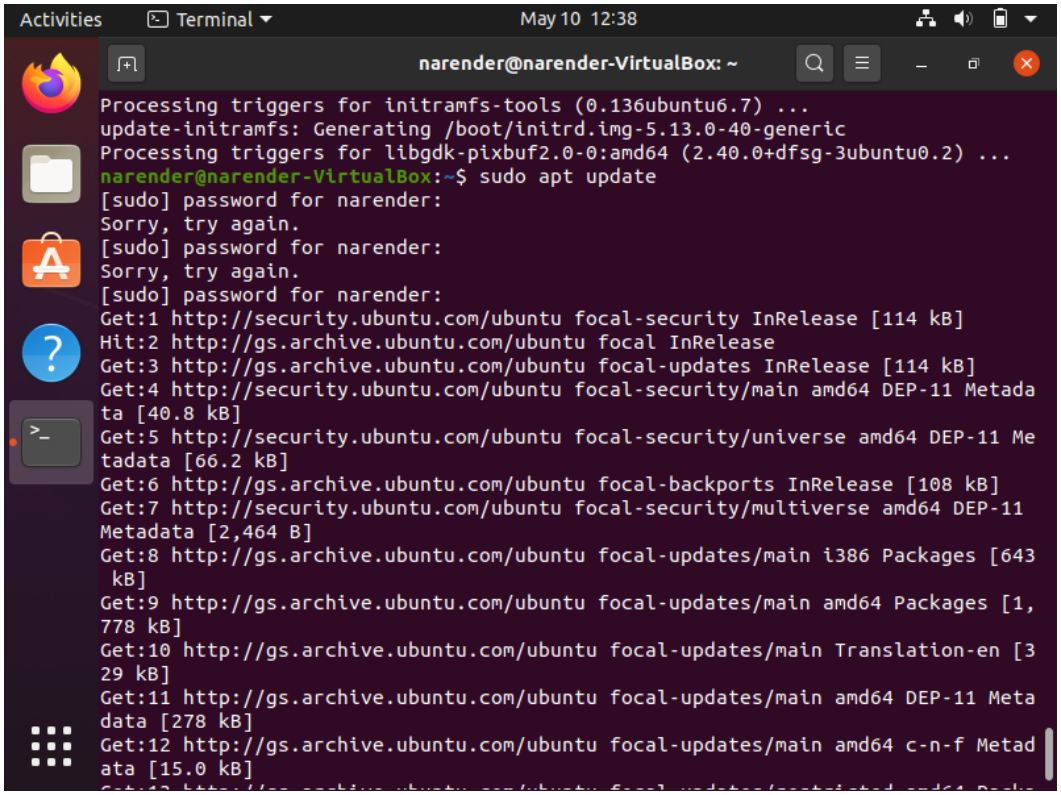
Steps for Installation:

1. Prerequisites:

sudo apt upgrade

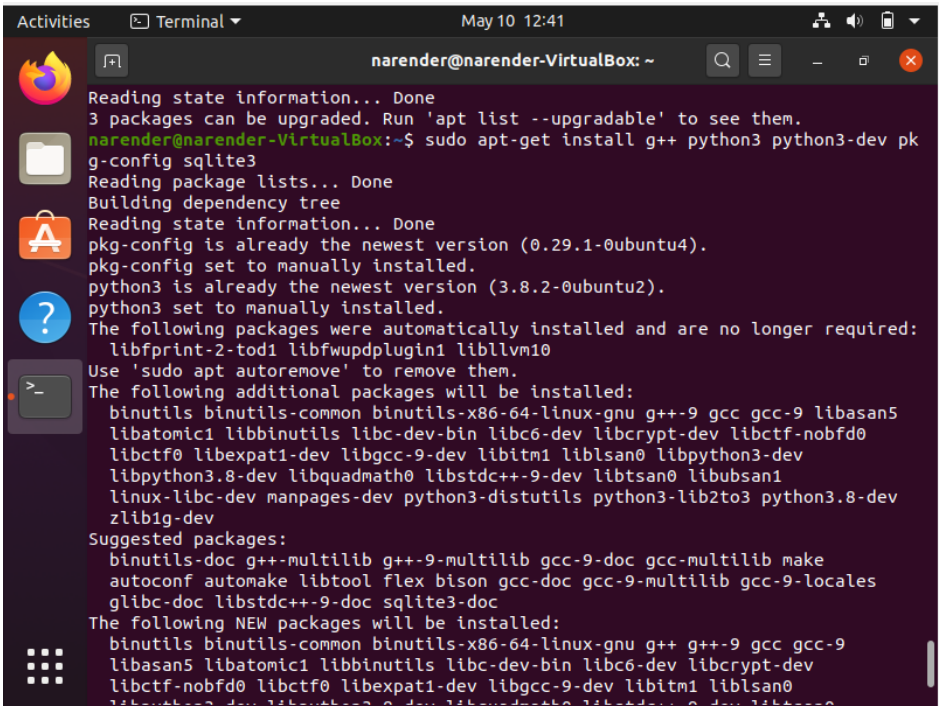


sudo apt update



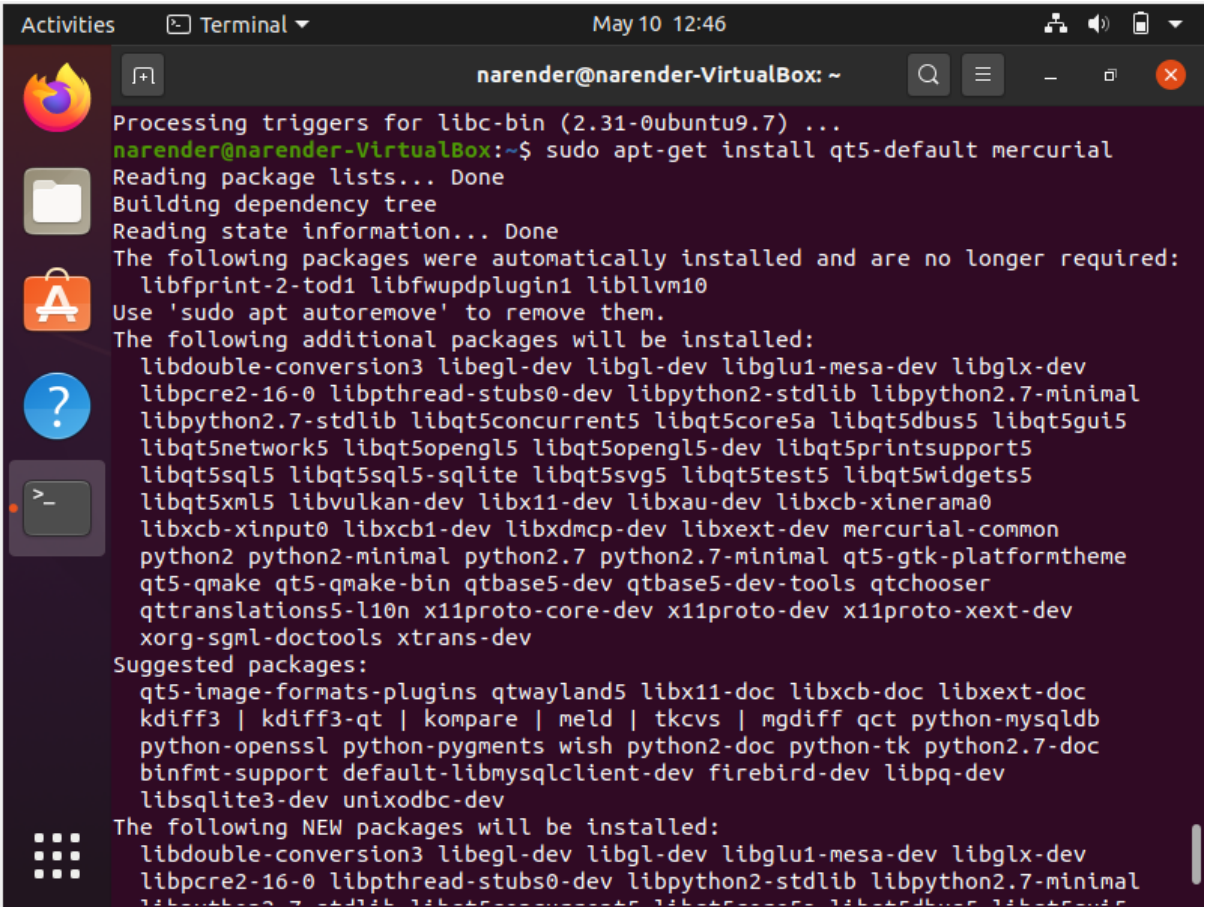
2. Minimal requirements for Python API users

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



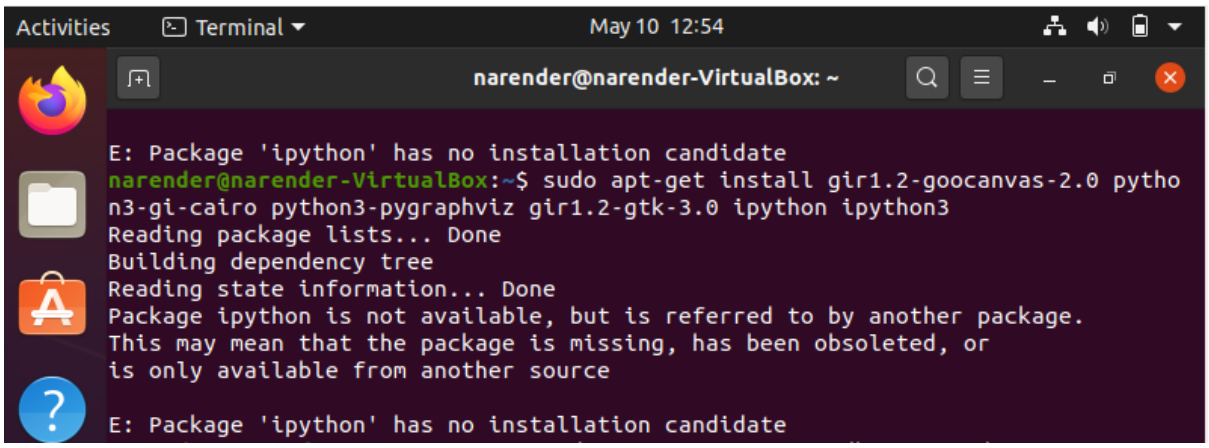
3. qt5 development tools are needed for Netanim animator:

sudo apt-get install qt5-default mercurial



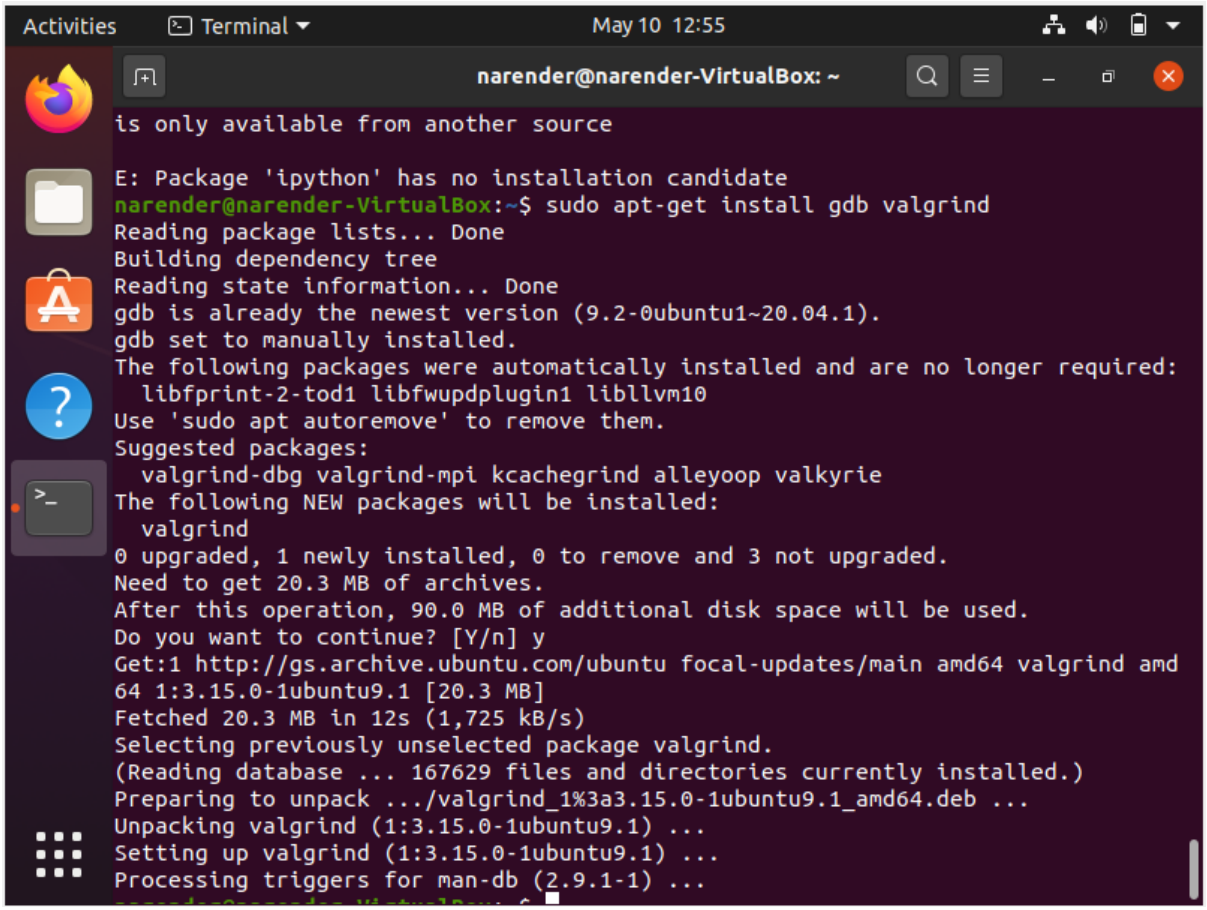
4. ns-3-pyviz visualizer:

sudo apt-get install gir1.2-goocanvas-2.0 python3-gi python3-gi-cairo python3pygraphviz gir1.2-gtk-3.0 ipython ipython3



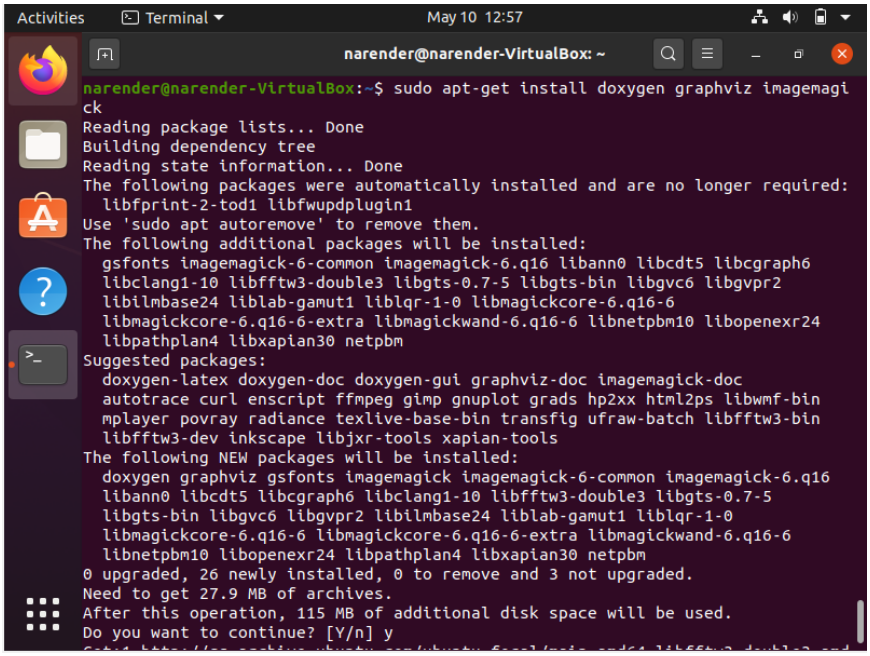
5. Debugging:

sudo apt-get install gdb valgrind

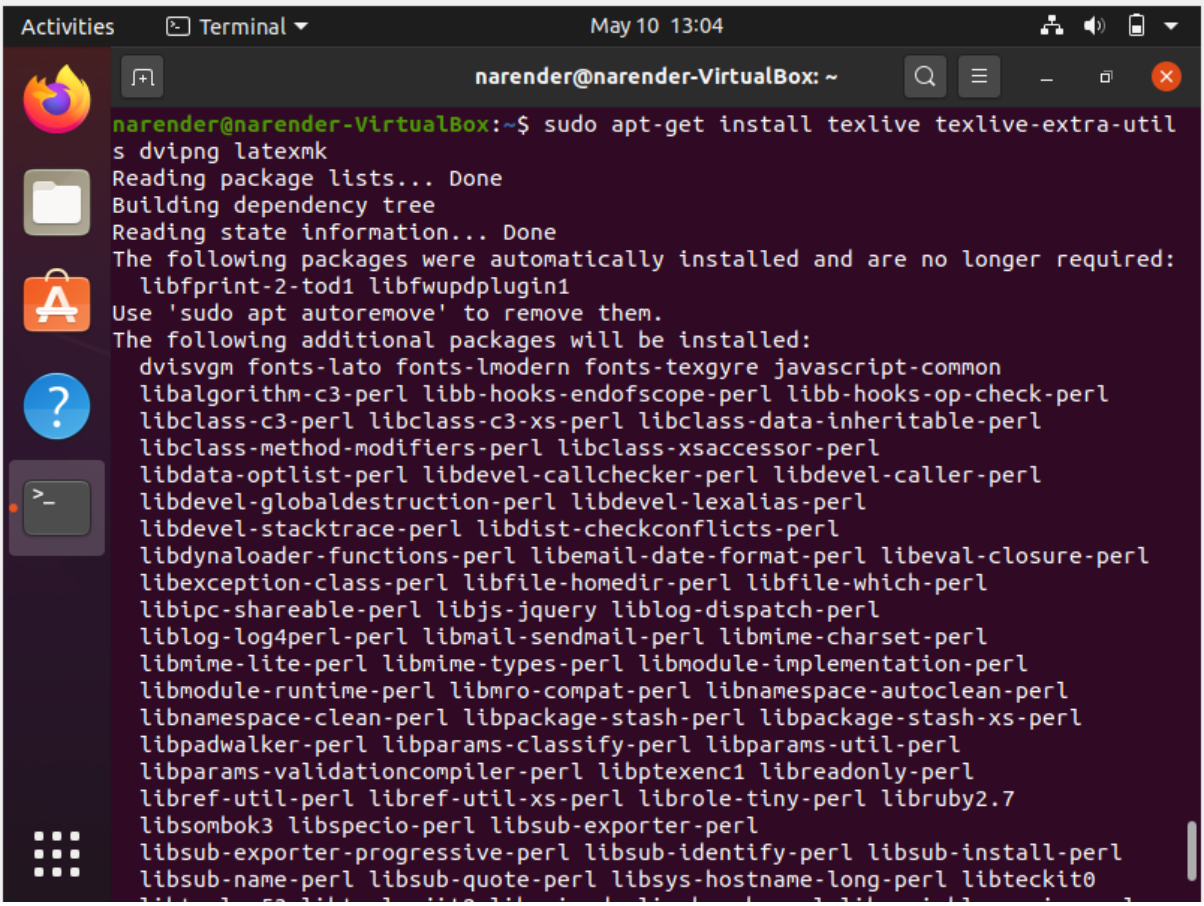


6. Doxygen and related inline documentation:

sudo apt-get install doxygen graphviz imagemagick

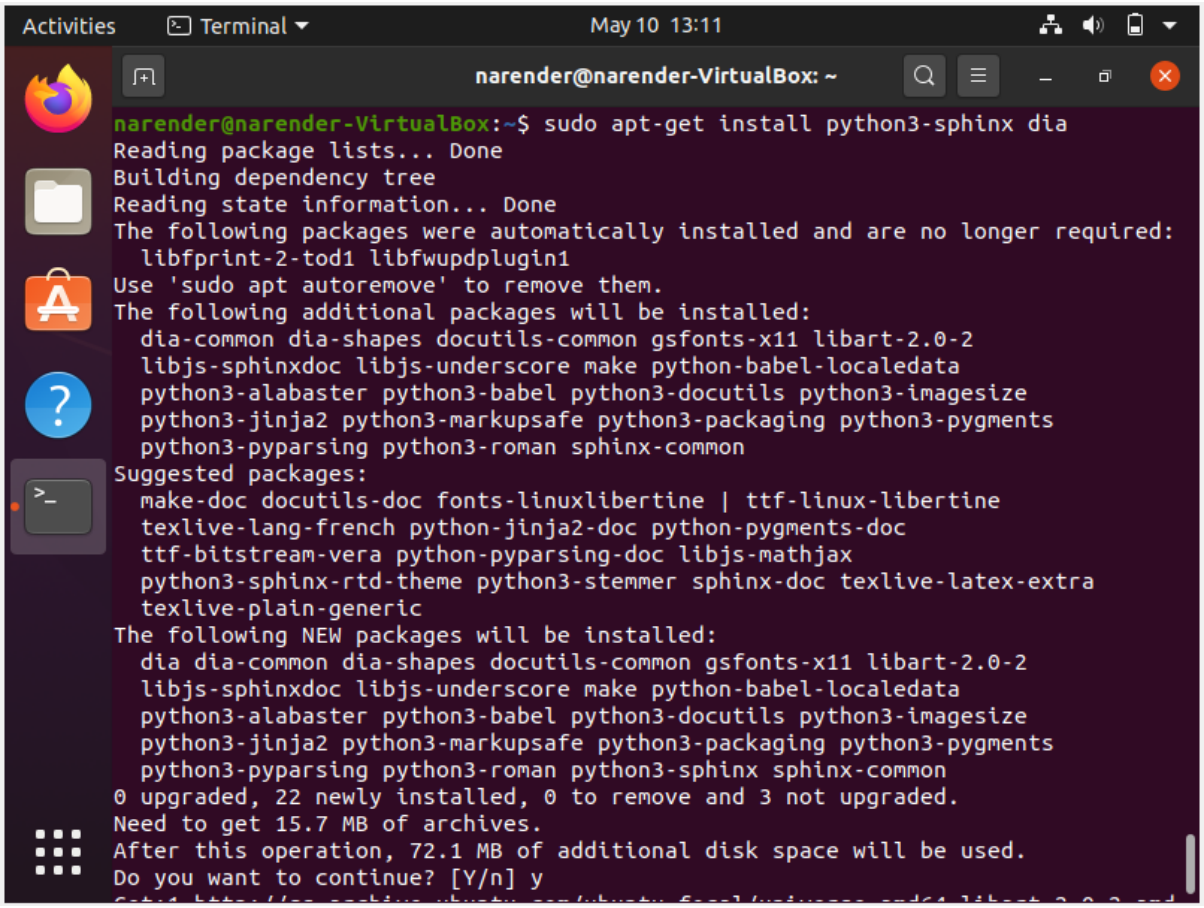


sudo apt-get install texlive texlive-extra-utils texlive-latex-extra texlive-font-utils dvipng latexmk



7. 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):

sudo apt-get install python3-sphinx dia



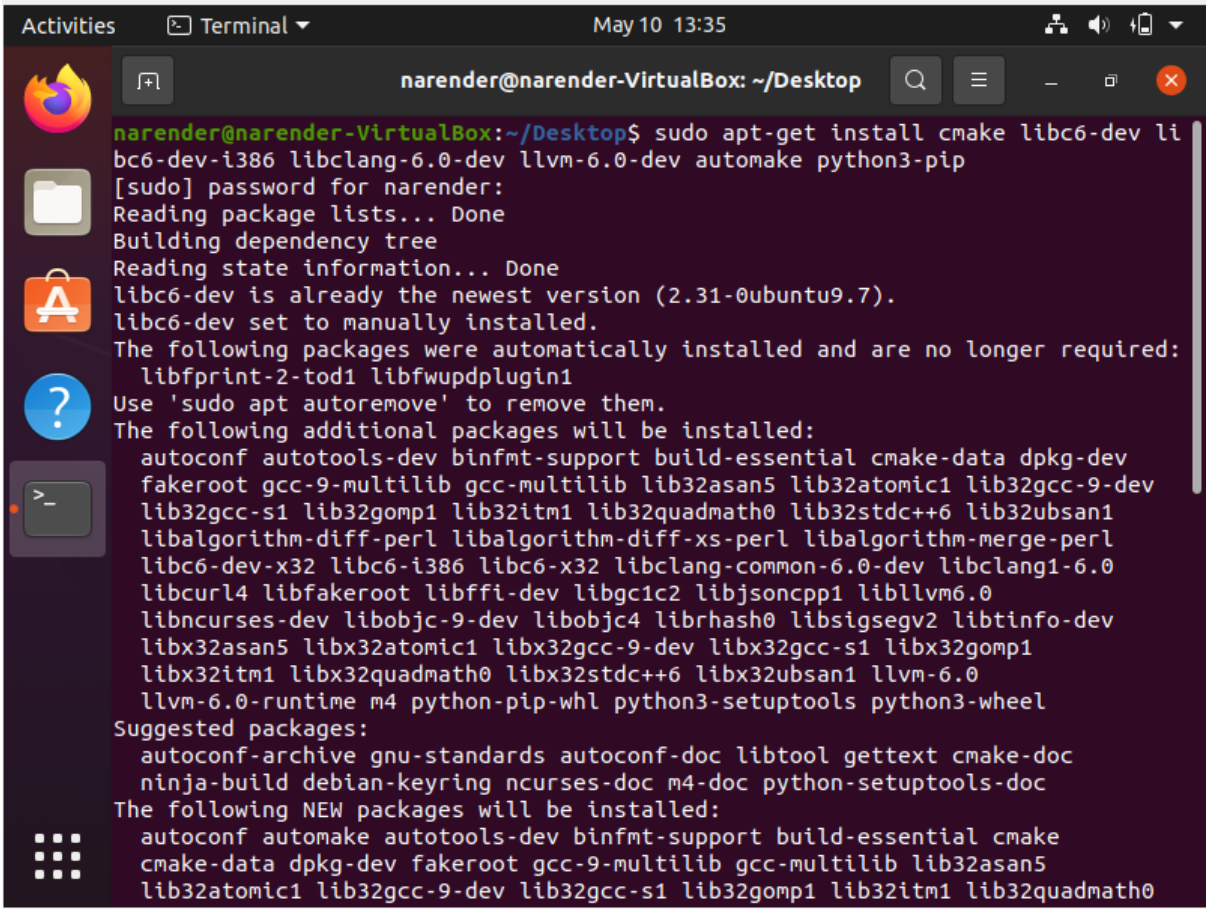
8. To read pcap packet traces:

sudo apt-get install tcpdump

```
narender@narender-VirtualBox:~$ sudo apt-get install tcpdump
Reading package lists... Done
Building dependency tree
Reading state information... Done
tcpdump is already the newest version (4.9.3-4ubuntu0.1).
tcpdump set to manually installed.
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 libfwupdplugin1
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
narender@narender-VirtualBox:~$
```

9. Support for generating modified python bindings:

sudo 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

```
narender@narender-VirtualBox:~/Desktop$ python3 -m pip install --user cxxfilt
Collecting cxxfilt
  Downloading cxxfilt-0.3.0-py2.py3-none-any.whl (4.6 kB)
Installing collected packages: cxxfilt
Successfully installed cxxfilt-0.3.0
```

10. ns-3 Build and test:

Go to ns-3.33 folder

cd workspace/ ns-allinone-3.33/

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

wget -c https://www.nsnam.org/releases/ns-allinone-3.33.tar.bz2

```
narender@narender-VirtualBox:~/Desktop$ wget -c https://www.nsnam.org/releases/
ns-allinone-3.33.tar.bz2
--2022-05-10 13:58:09--  https://www.nsnam.org/releases/ns-allinone-3.33.tar.bz
2
Resolving www.nsnam.org (www.nsnam.org)... 143.215.76.161
Connecting to www.nsnam.org (www.nsnam.org)|143.215.76.161|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28412785 (27M) [application/x-bzip2]
Saving to: ‘ns-allinone-3.33.tar.bz2’

ns-allinone-3.33.ta 100%[==================>]  27.10M  1.98MB/s    in 17s

2022-05-10 13:58:27 (1.59 MB/s) - ‘ns-allinone-3.33.tar.bz2’ saved [28412785/28
412785]

narender@narender-VirtualBox:~/Desktop$
```

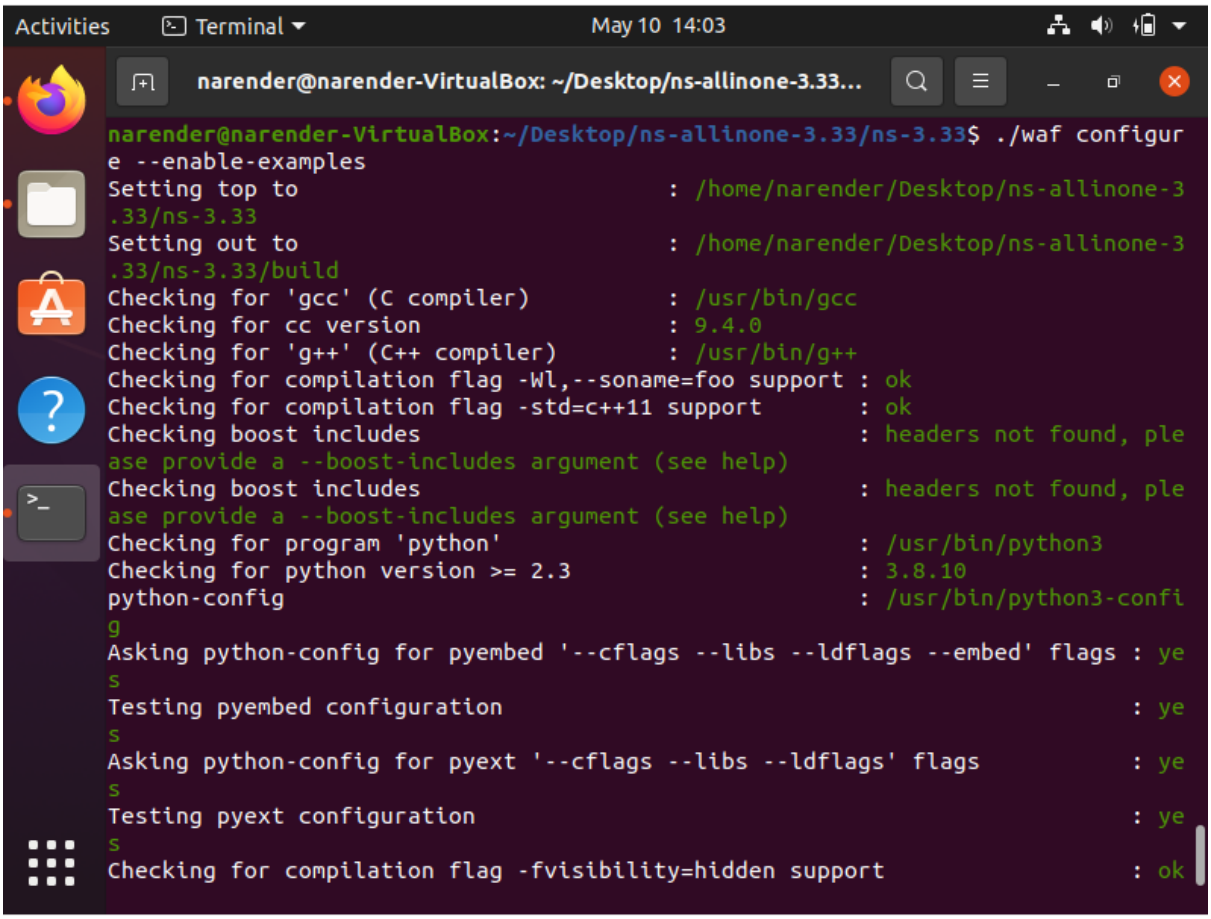
tar -xvjf ns-allinone-3.33.tar.bz2

```
narender@narender-VirtualBox:~/Desktop$ tar -xvjf ns-allinone-3.33.tar.bz2
ns-allinone-3.33/ns-3.33/wscript
ns-allinone-3.33/ns-3.33/utils.py
ns-allinone-3.33/ns-3.33/wutils.py
ns-allinone-3.33/ns-3.33/test.py
ns-allinone-3.33/ns-3.33/LICENSE
ns-allinone-3.33/ns-3.33/CHANGES.html
ns-allinone-3.33/ns-3.33/Makefile
ns-allinone-3.33/ns-3.33/README.md
ns-allinone-3.33/ns-3.33/waf
ns-allinone-3.33/ns-3.33/testpy.supp
ns-allinone-3.33/ns-3.33/AUTHORS
ns-allinone-3.33/ns-3.33/waf.bat
ns-allinone-3.33/ns-3.33/RELEASE_NOTES
ns-allinone-3.33/ns-3.33/CONTRIBUTING.md
ns-allinone-3.33/ns-3.33/VERSION
ns-allinone-3.33/ns-3.33/contrib/wscript
ns-allinone-3.33/ns-3.33/examples/socket/wscript
ns-allinone-3.33/ns-3.33/examples/socket/socket-options-ipv6.cc
ns-allinone-3.33/ns-3.33/examples/socket/socket-options-ipv4.cc
ns-allinone-3.33/ns-3.33/examples/socket/socket-bound-tcp-static-routing.cc
ns-allinone-3.33/ns-3.33/examples/socket/socket-bound-static-routing.cc
ns-allinone-3.33/ns-3.33/examples/traffic-control/wscript
ns-allinone-3.33/ns-3.33/examples/traffic-control/red-vs-nlred.cc
ns-allinone-3.33/ns-3.33/examples/traffic-control/cobalt-vs-codel.cc
```

cd ns-allinone-3.33/ns-3.33/

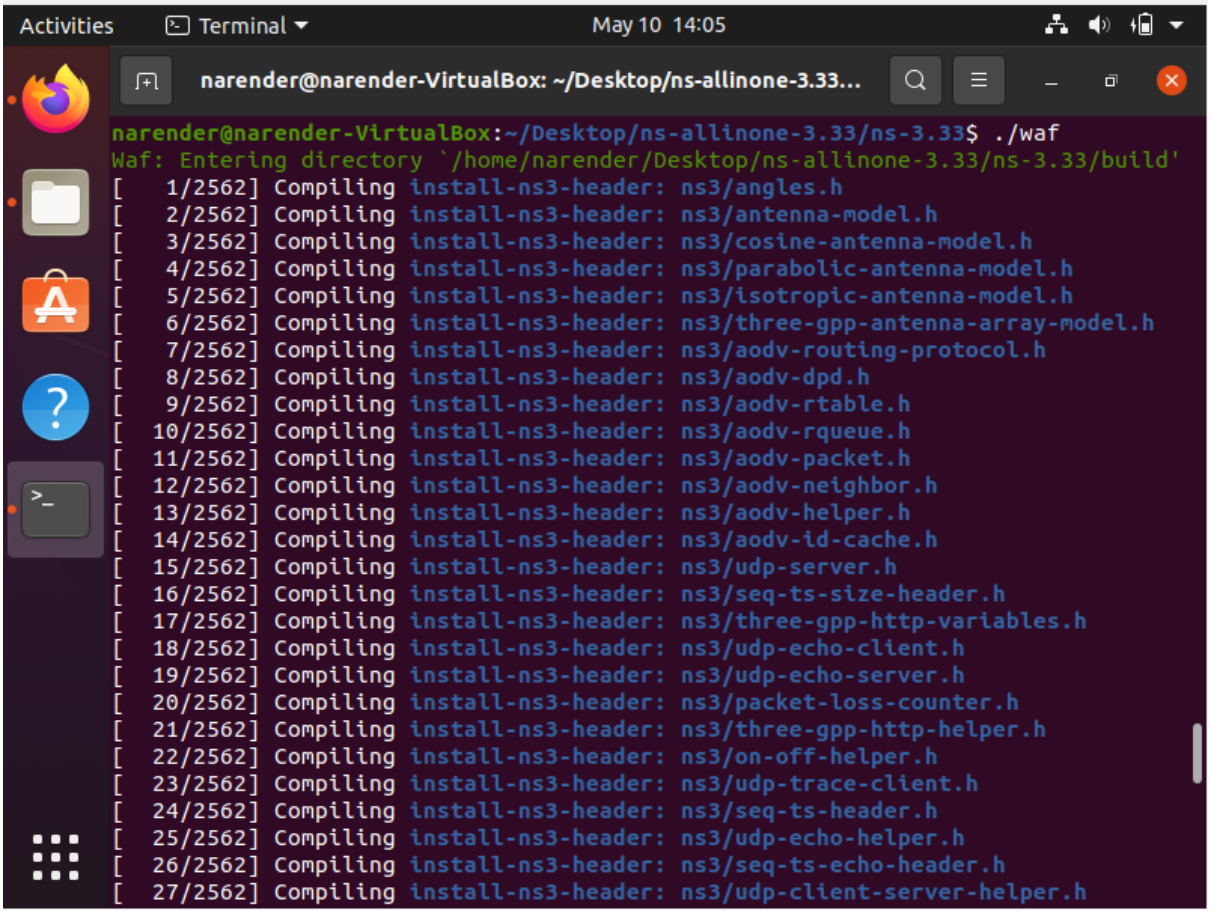
```
narender@narender-VirtualBox:~/Desktop$ cd ns-allinone-3.33/ns-3.33/
narender@narender-VirtualBox:~/Desktop/ns-allinone-3.33/ns-3.33$
```

 # Configure the installation
./waf configure --enable-examples

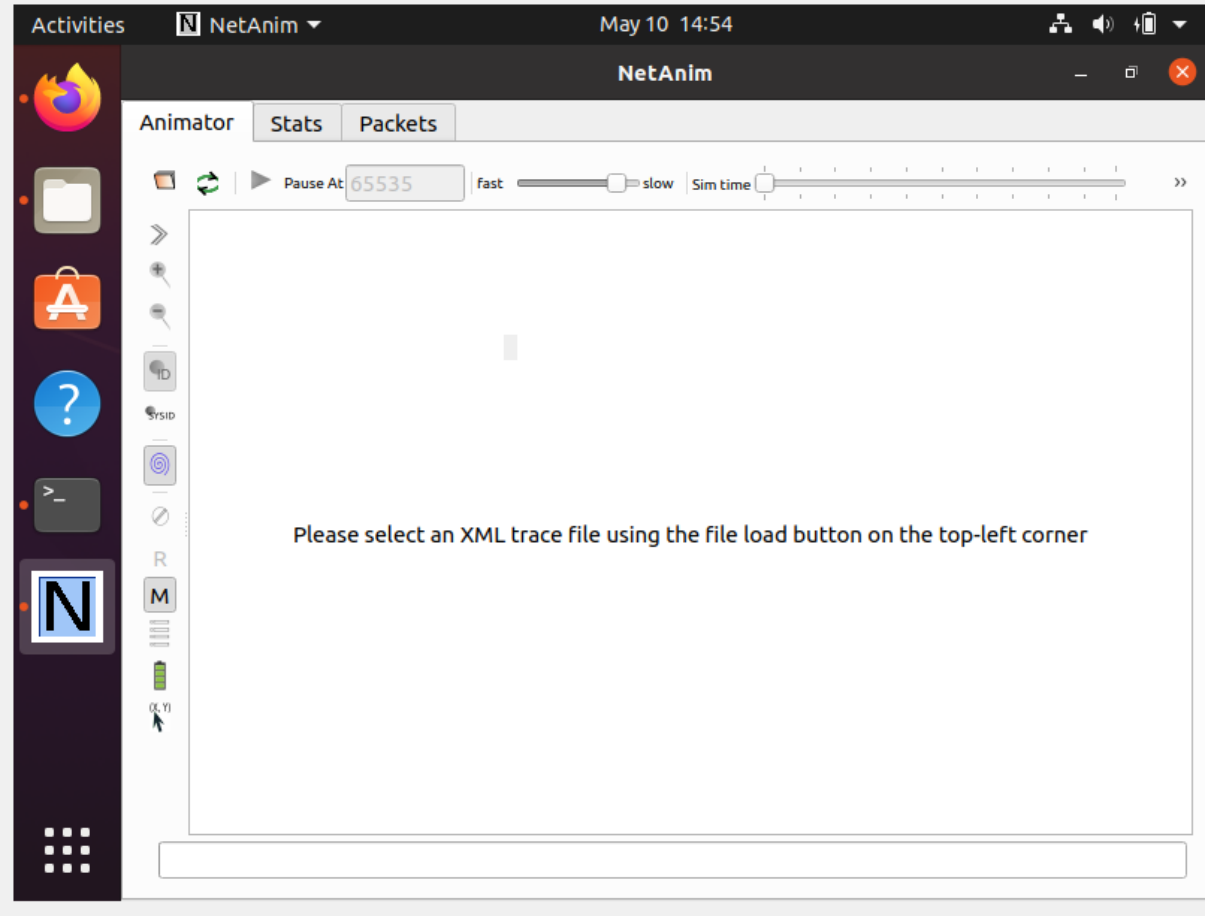


 # Build ns-3 installation

```
./waf
```

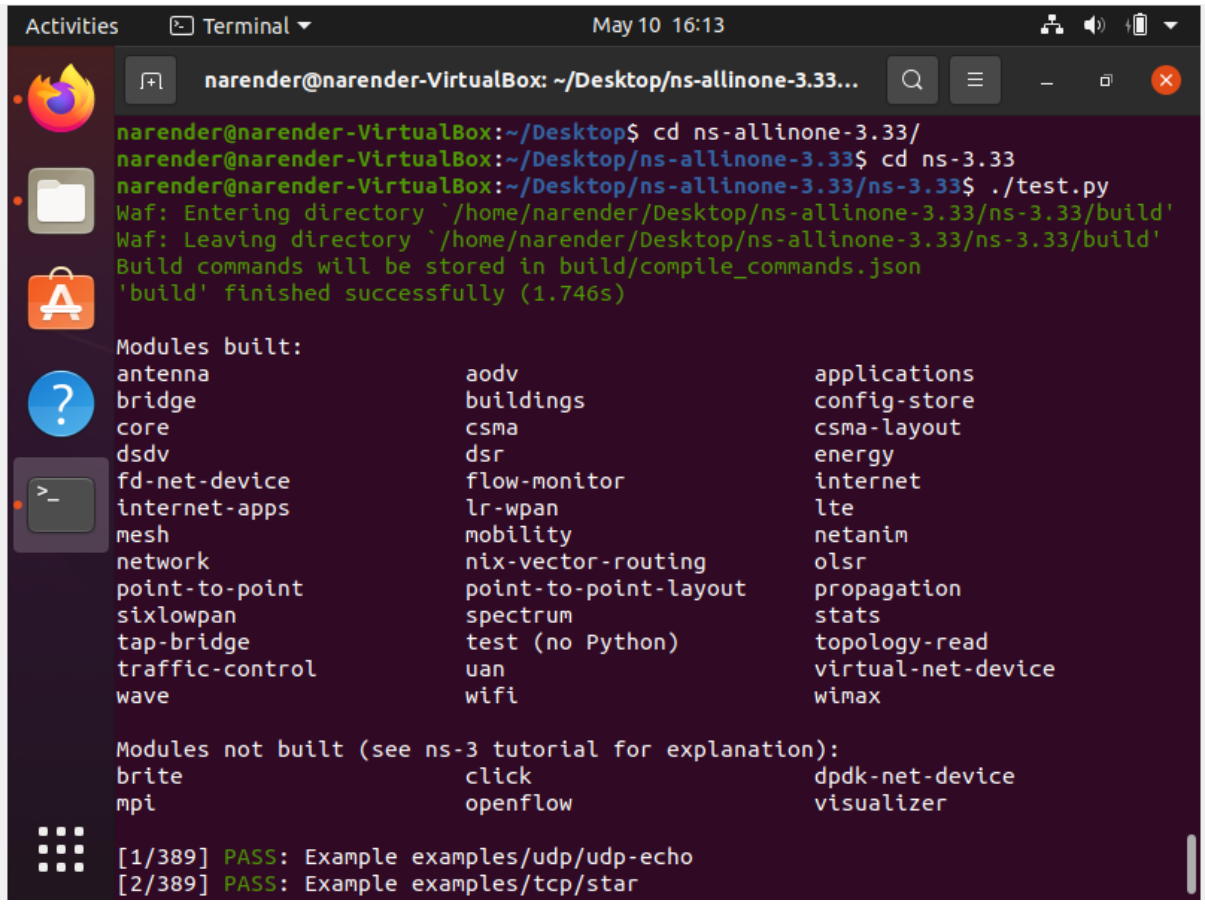



```
# to check whether installation was a success
./waf --run hello-simulator

# navigate to netanim dir.
cd ns-allinone-3.33/netanim-3.108/
# configure the build
make clean
# compile the build
qmake NetAnim.pro
# build netanim installation
make
# to execute NetAnim
./NetAnim
```



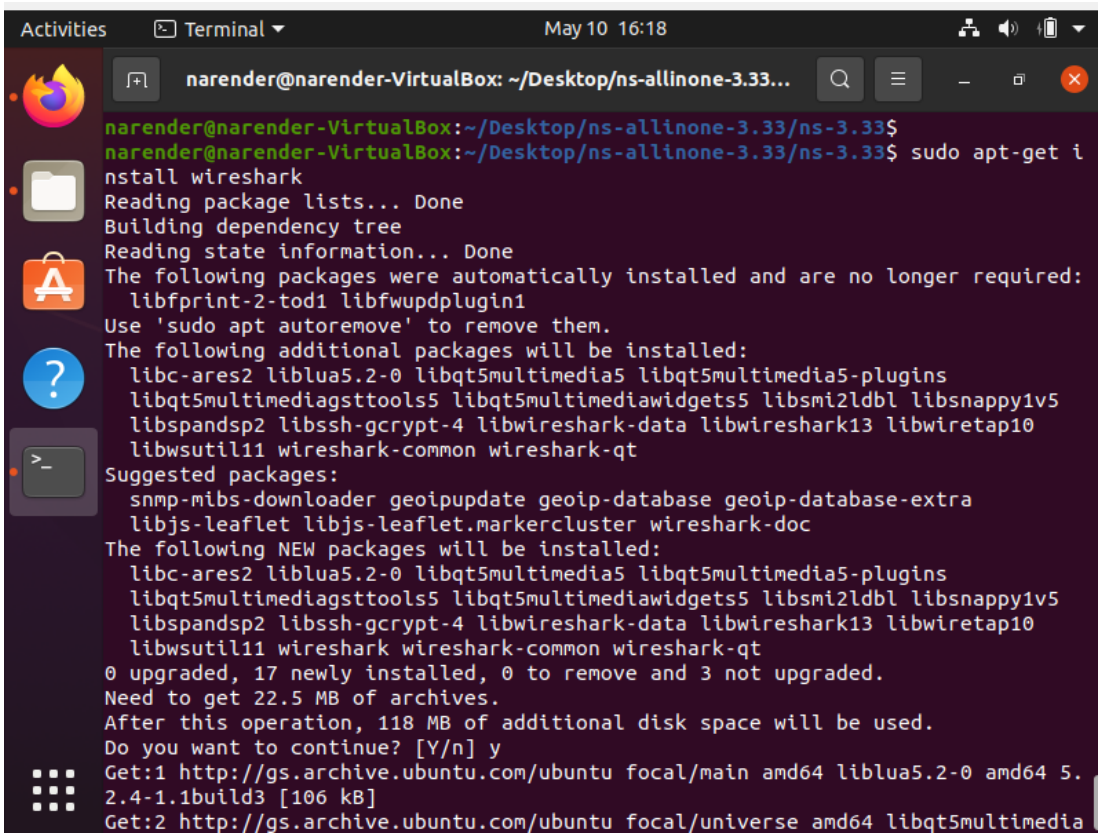
11. Test the NS3 build and installation success by running test.py in the ns directory using the following commands:

cd workspace/ ns-allinone-3.33/ ns-3.33
./test.py



12. Install WireShark

sudo apt-get install wireshark



Conclusion:

From this practical, I have Successfully installed Ns-3, NetAnim and WireShark.

AIM: To simulate traffic between two nodes (point to point).

THEORY:

What is Node?

a. In Internet jargon, a computing device that connects to a network is called a host or sometimes an end system.
b. Because ns-3 is a network simulator, not specifically an Internet simulator, it does not use the term host since it is closely associated with the Internet and its protocols.
c. Instead, it uses a more generic term also used by other simulators that originates in Graph Theory the node.
d. In ns-3 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.
e. Thin Node as a computer to which functionality can be added. One adds things like applications, protocol stacks and peripheral cards with their associated drivers to enable the computer to do useful work.

Netdevice:

i. Net device abstraction covers both the software driver and the simulated hardware.
ii. A net device is installed in a Node in order to enable the Node to communicate with other Nodes in the simulation via Channels.
iii. Just as in a real computer, a Node may be connected to more than one Channel via multiple NetDevices.
iv. The net device abstraction is represented in C++ by the class NetDevice.
v. The NetDevice class provides methods for managing connections to Node and Channel objects.

Classes used in code:

Following different classes are used in code:

a. Node class:

i. In ns-3 the basic computing device abstraction is called the node. This abstraction is represented in C++ by the class Node.
ii. The Node class provides methods for managing the representations of computing devices in simulations.

b. Application class:

i. In ns-3 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. ii. The Application class provides methods for managing the representations of 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.

c. Channel class:

i. The basic communication subnetwork abstraction is called the channel and is represented in C++ by the class Channel.
ii. The Channel class provides methods for managing communication subnetwork objects and connecting nodes to them.

d. NetDevice Class:

i. Net device abstraction covers both the software driver and the simulated hardware. The net device abstraction is represented in C++ by the class NetDevice. ii. 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.

e. NodeContainer Class:

i. Typically, ns-3 helpers operate on more than one node at a time. For example, a device helper may want to install devices on a large number of similar nodes.
ii. The helper Install methods usually take a NodeContainer as a parameter. NodeContainers hold the multiple Ptr<Node> which are used to refer to the nodes.

f. PointToPointHelper Class:

i. PointToPointNetDevice class specializes the NetDevice abstract base class.
ii. Together with a PointToPointChannel the class models, with some level of abstraction, a generic point-to-point or serial link. Key parameters or objects that can be specified for this device include a queue, data rate, and interframe transmission gap.

SOURCE CODE:

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

//Use ns3 namespace
```

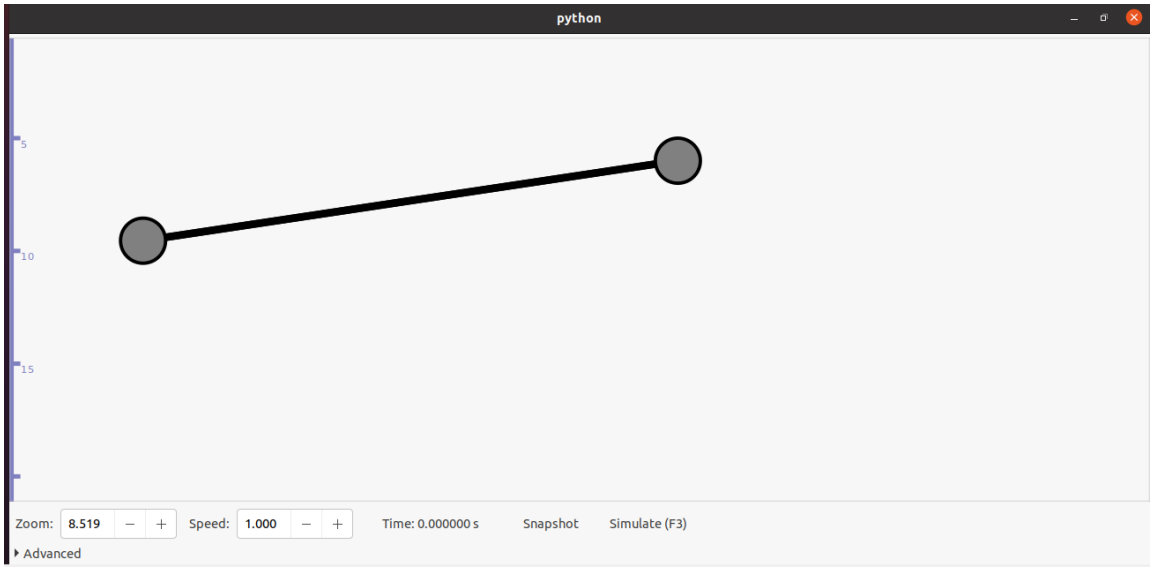
```
using namespace ns3;

// Enable log for this program

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

// Main function

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

{

// Enable this program to read and parse command line arguments

CommandLine cmd;

cmd.Parse (argc, argv);

// Enable Log of echo applications

LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);

LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

// Create nodes

NodeContainer nodes;

nodes.Create (2);

// Create a point-to-point channel and configure its attributes

PointToPointHelper pointToPoint;

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

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

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

// Install network stack on nodes

InternetStackHelper stack;

stack.Install(nodes);

// Set network address and subnet mask

Ipv4AddressHelper address;
```

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

// Assign IP address to every interface

Ipv4InterfaceContainer interfaces = address.Assign(devices);

// Configure a Server application

UdpEchoServerHelper echoServer(9);

// Install Server application on a specific node

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

// Set start and stop time for Server application

serverApps.Start (Seconds(1.0));

serverApps.Stop (Seconds(10.0));

//Configure a Client application

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

echoClient.SetAttribute ("MaxPackets", UintegerValue (5));

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

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

// Install Client application on a specific node

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

clientApps.Start (Seconds(2.0));

clientApps.Stop (Seconds(10.0));

// Run the simulation

pointToPoint.EnablePcapAll("first");

MobilityHelper mobility;

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

mobility.Install(nodes);

AnimationInterface anim("P2narender.xml");

AnimationInterface::SetConstantPosition (nodes.Get(0), 10, 25);
```

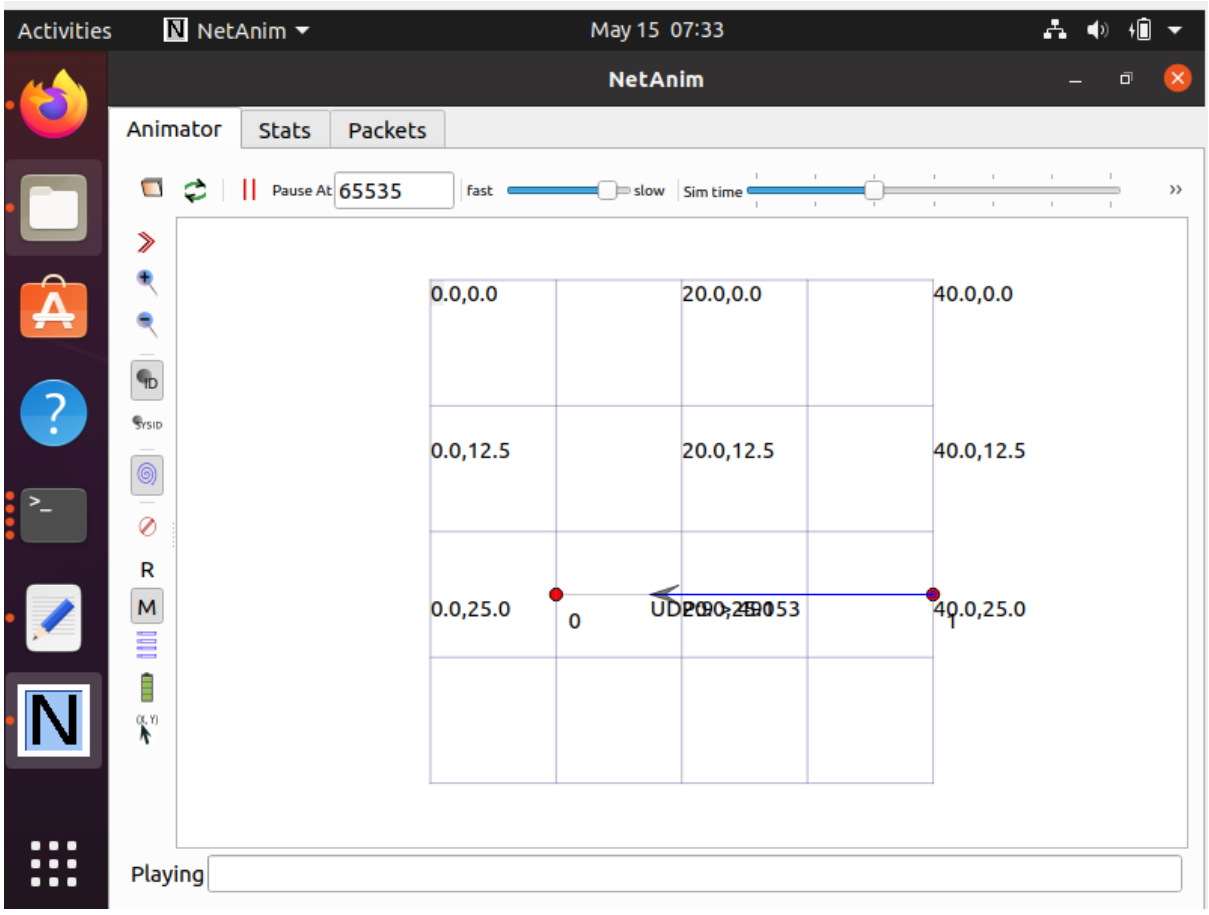
```
AnimationInterface ::SetConstantPosition(nodes.Get(1), 40,25);

anim.EnablePacketMetadata(true);

Simulator::Run ();

Simulator::Destroy ();

return 0;

}
```

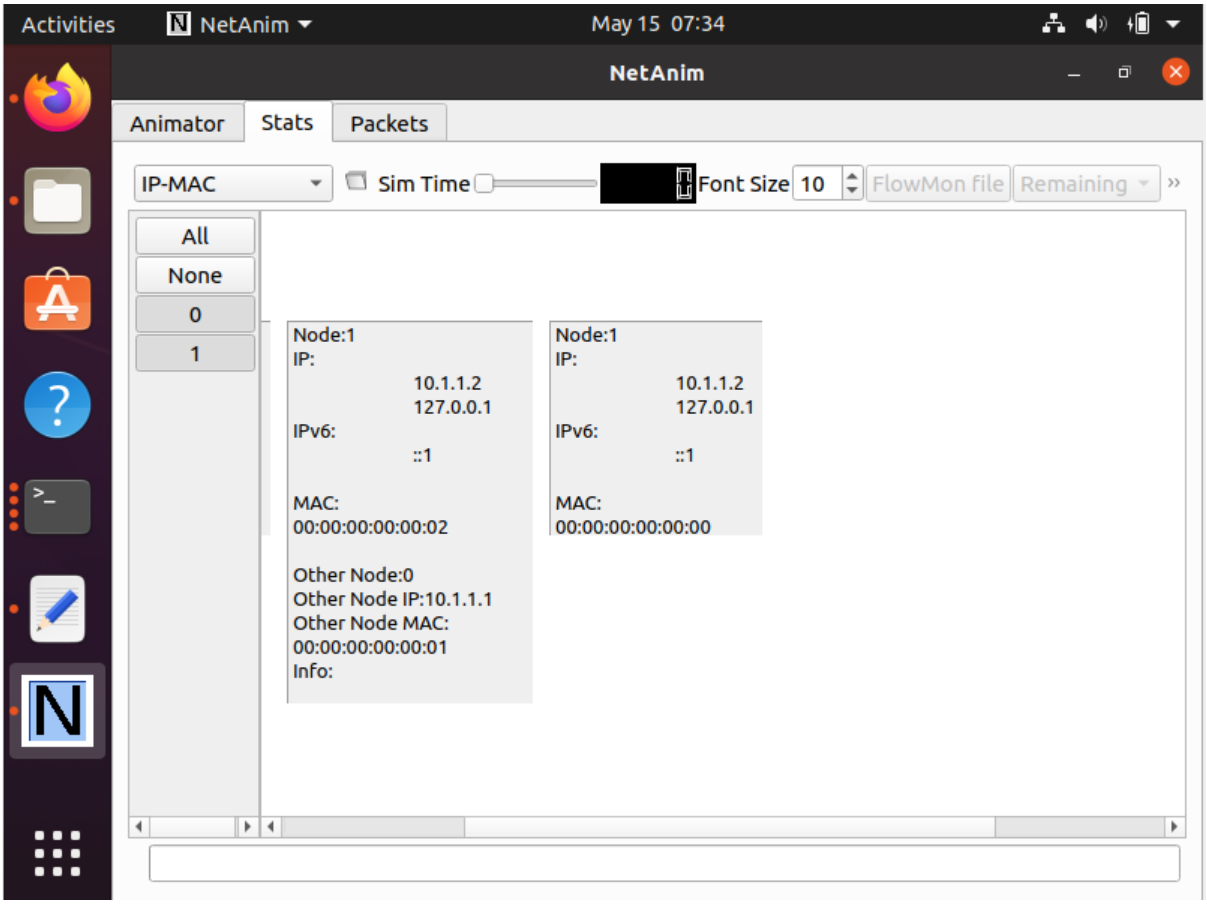
OUTPUT:

```
narender@narender-VirtualBox:~/Desktop/ns-allinone-3.33/ns-3.33$ ./waf --run sc
ratch/p2.cc
Waf: Entering directory `/home/narender/Desktop/ns-allinone-3.33/ns-3.33/build'
Waf: Leaving directory `/home/narender/Desktop/ns-allinone-3.33/ns-3.33/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.910s)
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
At time +3s client sent 1024 bytes to 10.1.1.2 port 9
At time +3.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time +3.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time +3.00737s client received 1024 bytes from 10.1.1.2 port 9
At time +4s client sent 1024 bytes to 10.1.1.2 port 9
At time +4.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time +4.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time +4.00737s client received 1024 bytes from 10.1.1.2 port 9
At time +5s client sent 1024 bytes to 10.1.1.2 port 9
At time +5.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time +5.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time +5.00737s client received 1024 bytes from 10.1.1.2 port 9
At time +6s client sent 1024 bytes to 10.1.1.2 port 9
At time +6.00369s server received 1024 bytes from 10.1.1.1 port 49153
At time +6.00369s server sent 1024 bytes to 10.1.1.1 port 49153
At time +6.00737s client received 1024 bytes from 10.1.1.2 port 9
narender@narender-VirtualBox:~/Desktop/ns-allinone-3.33/ns-3.33$
```

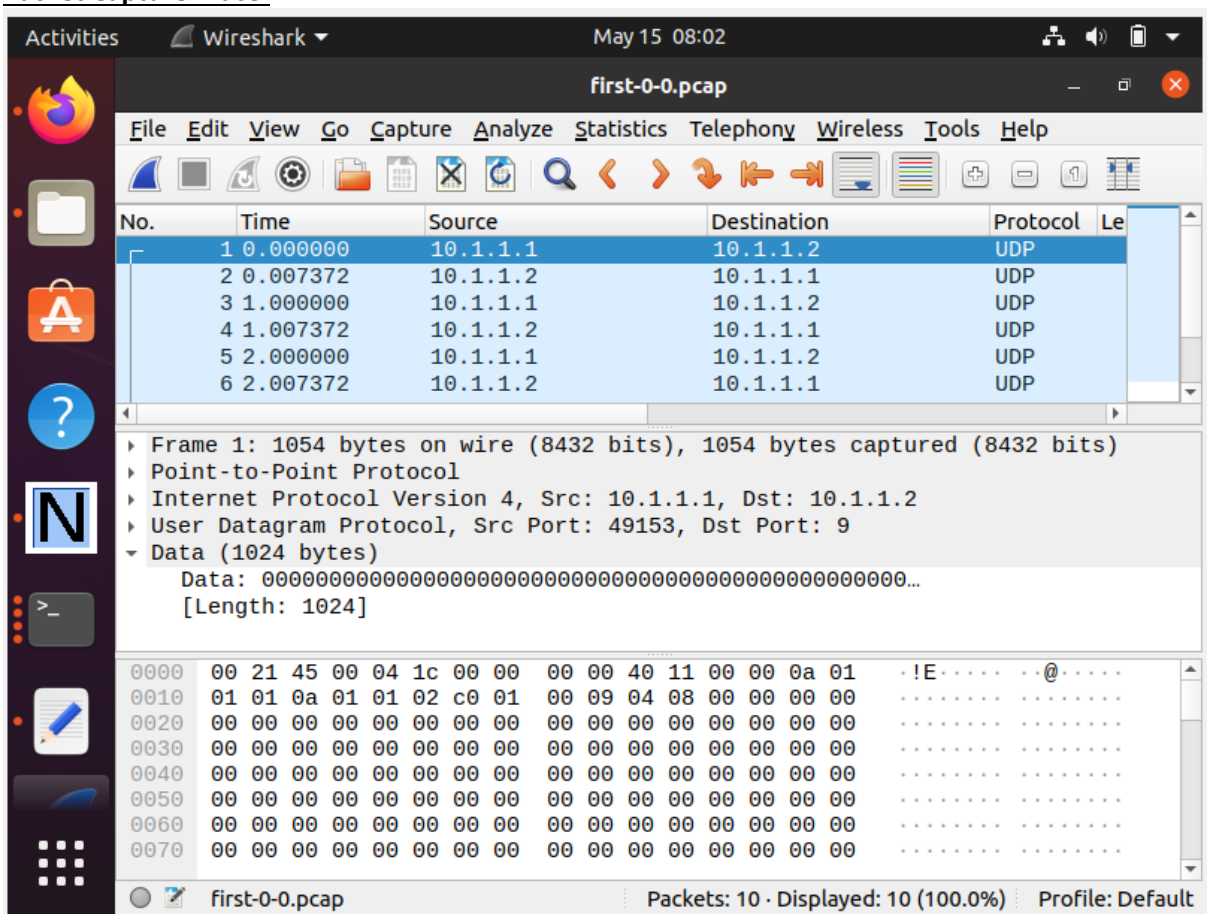

python
Zoom: 8.519
Speed: 1.000
Time: 0.000000 s
Snapshot
Simulate (F3)
Advanced

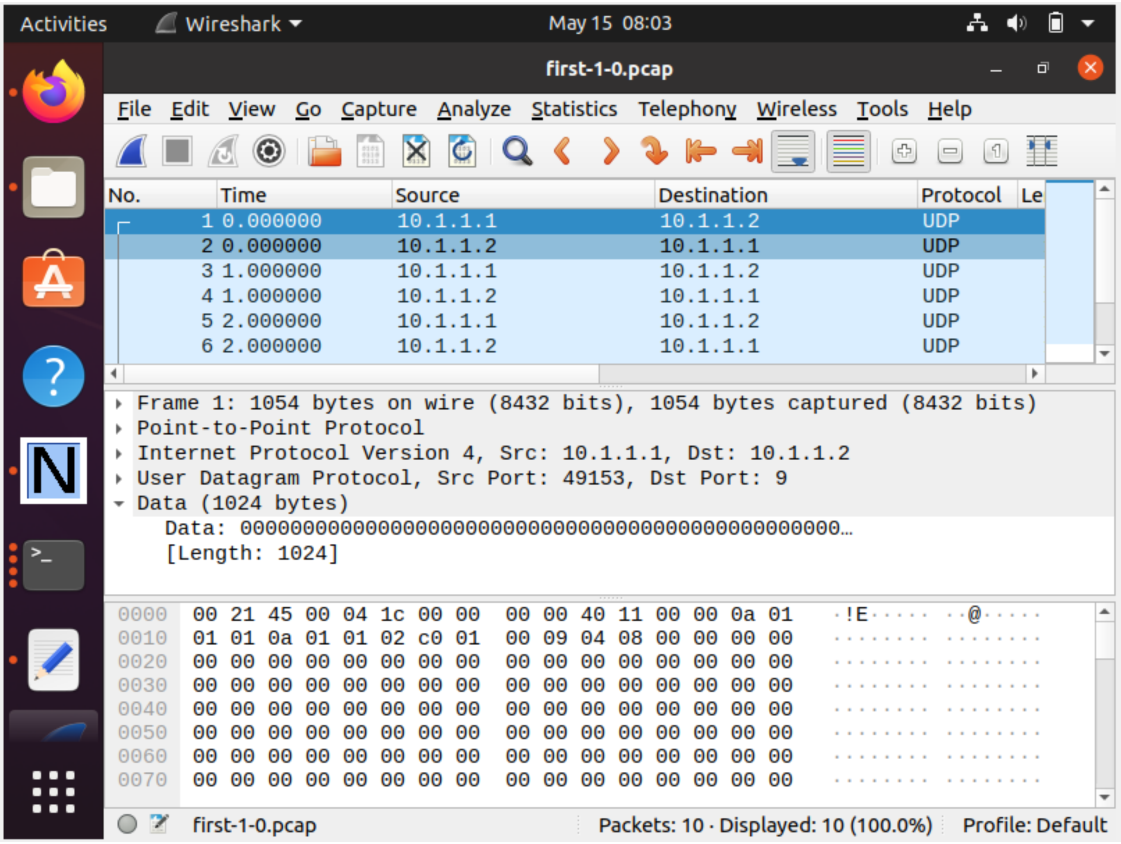
NetAnim:


Activities
NetAnim
May 15 07:33
NetAnim
Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
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
40.0,25.0
0
1
R
M
Playing

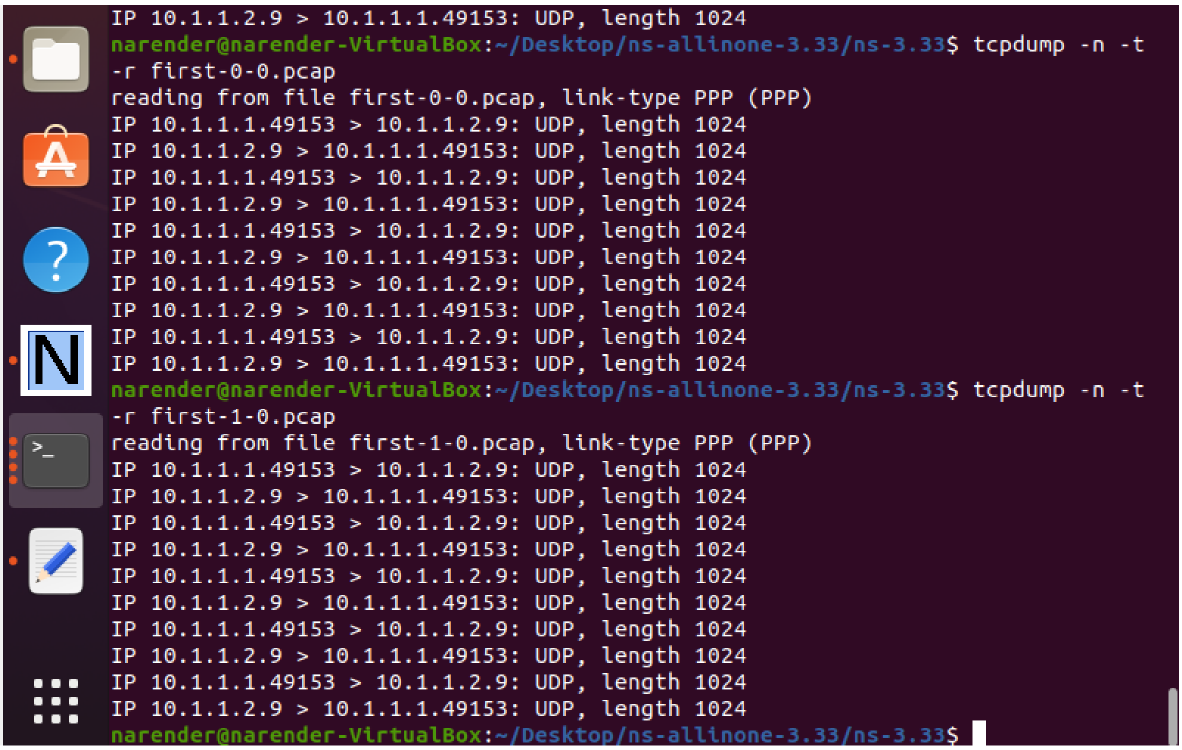


Packet Capture Trace:





Using TcpDump:



CONCLUSION:

From this practical, I have learned about point-to-point simulation in ns3.

AIM: Program to simulate star topology.

THEORY:

Understanding Network Topology:

Network topology is the arrangement of the elements (links, nodes, etc.) of a communication network. Network topology can be used to define or describe the arrangement of various types of telecommunication networks, including command and control radio networks, industrial field busses and computer networks.

Network topology is the topological structure of a network and may be depicted physically or logically. It is an application of graph theory wherein communicating devices are modelled as nodes and the connections between the devices are modelled as links or lines between the nodes.

Physical topology is the placement of the various components of a network (e.g., device location and cable installation), while logical topology illustrates how data flows within a network. Distances between nodes, physical interconnections, transmission rates, or signal types may differ between two different networks, yet their logical topologies may be identical.

A network's physical topology is a particular concern of the physical layer of the OSI model.

Examples of network topologies are found in local area networks (LAN), a common computer network installation. Any given node in the LAN has one or more physical links to other devices in the network; graphically mapping these links results in a geometric shape that can be used to describe the physical topology of the network.

A wide variety of physical topologies have been used in LANs, including ring, bus, mesh and star. Conversely, mapping the data flow between the components determines the logical topology of the network. In comparison, Controller Area Networks, common in vehicles, are primarily distributed control system networks of one or more controllers interconnected with sensors and actuators over, invariably, a physical bus topology.

Star Topology:

Star topology is a network topology where each individual piece of a network is attached to a central node (often called a hub or switch). The attachment of these network pieces to the central component is visually represented in a form similar to a star.

Star topology is also known as a star network.

Star topologies are either active or passive networks, depending on the following:

- If the central node performs processes, such as data amplification or regeneration
- If the network actively controls data transit
- If the network requires electrical power sources.

Star topologies also may be implemented with Ethernet/cabled structures, wireless routers and/or other components. In many cases, the central hub is the server, and the additional nodes are clients.

Benefits of a star network topology include the following:

- Has the ability to limit the impact of a single failure. In star networks, a single unit is isolated by its relationship to the central hub, so that if a component goes down, it only affects that unit's local reach.
- Facilitates adding or removing individual components to and from a network, for the same reasons.

Disadvantages of star topology:

- May have a higher cost to implement, especially when using a switch or router as the central network device.
- The central network device determines the performance and number of nodes the network can handle.
- If the central computer, hub, or switch fails, the entire network goes down and all computers are disconnected from the network.

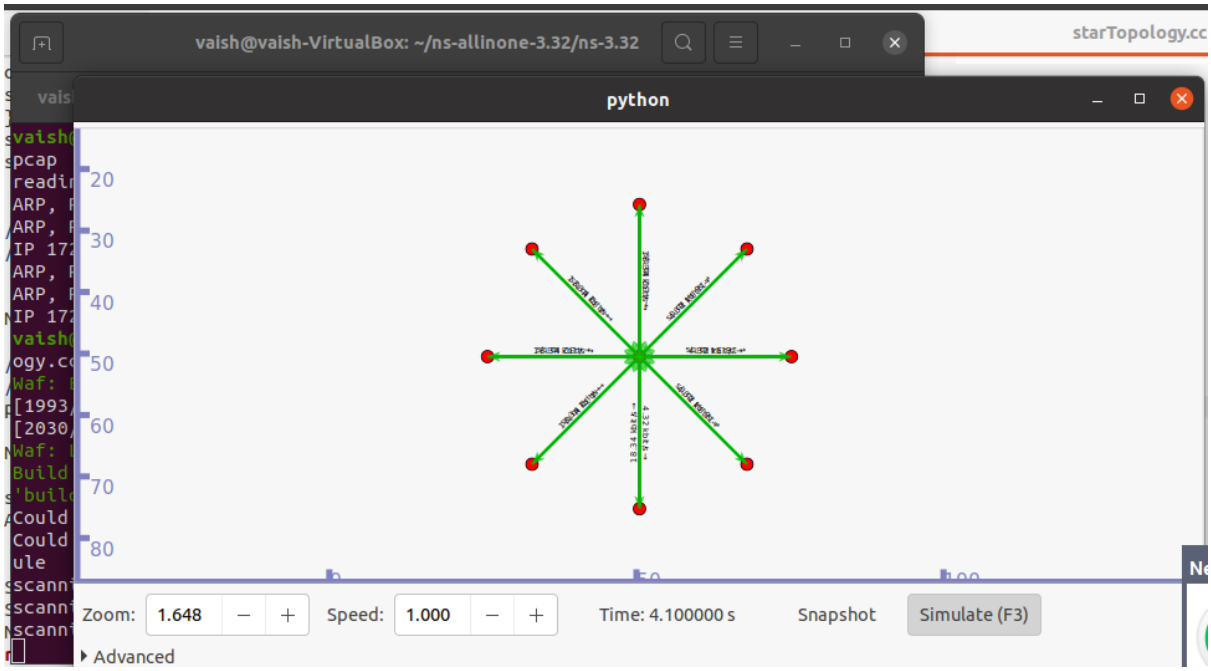
SOURCE CODE:

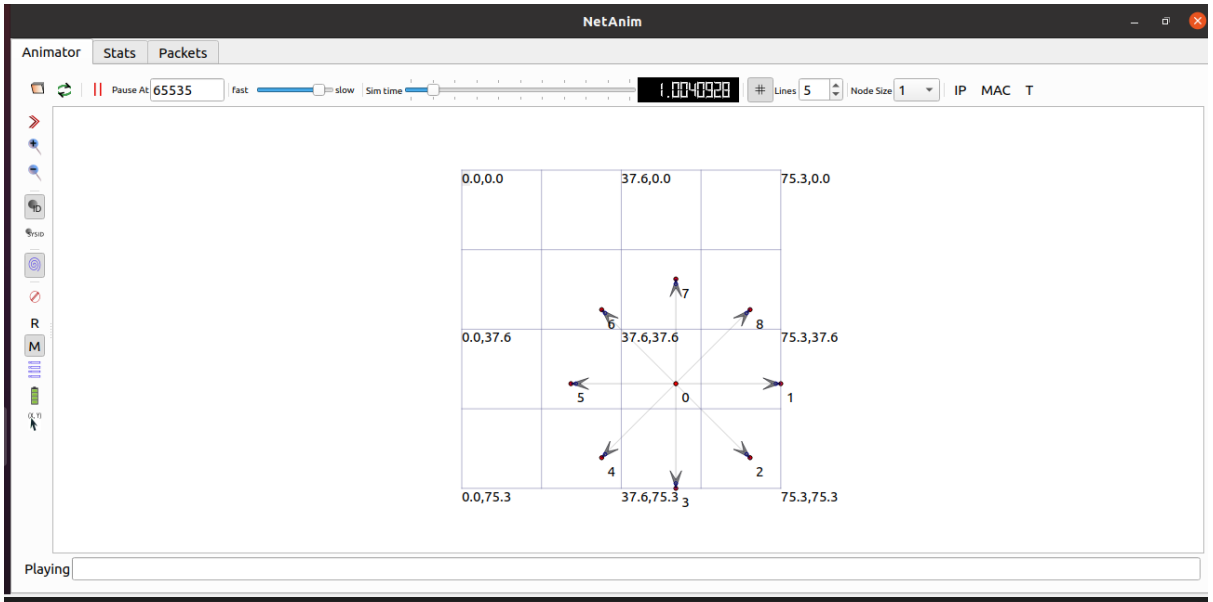
```
#include "ns3/core-module.h"
 #include "ns3/network-module.h"
 #include "ns3/netanim-module.h"
 #include "ns3/internet-module.h"
 #include "ns3/point-to-point-module.h"
 #include "ns3/applications-module.h"
 #include "ns3/point-to-point-layout-module.h"

 using namespace ns3;

 NS_LOG_COMPONENT_DEFINE ("Star");
 int main (int argc, char *argv[])
 {
  // Set up some default values for the simulation.
 Config::SetDefault ("ns3::OnOffApplication::PacketSize", UintegerValue (137));

  // ??? try and stick 15kb/s into the data rate
 Config::SetDefault ("ns3::OnOffApplication::DataRate", StringValue ("14kb/s"));

// Default number of nodes in the star. Overridable by command line argument.
   uint32_t nSpokes = 8;

  CommandLine cmd;
  cmd.AddValue ("nSpokes", "Number of nodes to place in the star", nSpokes);
  cmd.Parse (argc, argv);

  NS_LOG_INFO ("Build star topology.");
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
  PointToPointStarHelper star (nSpokes, pointToPoint);
```

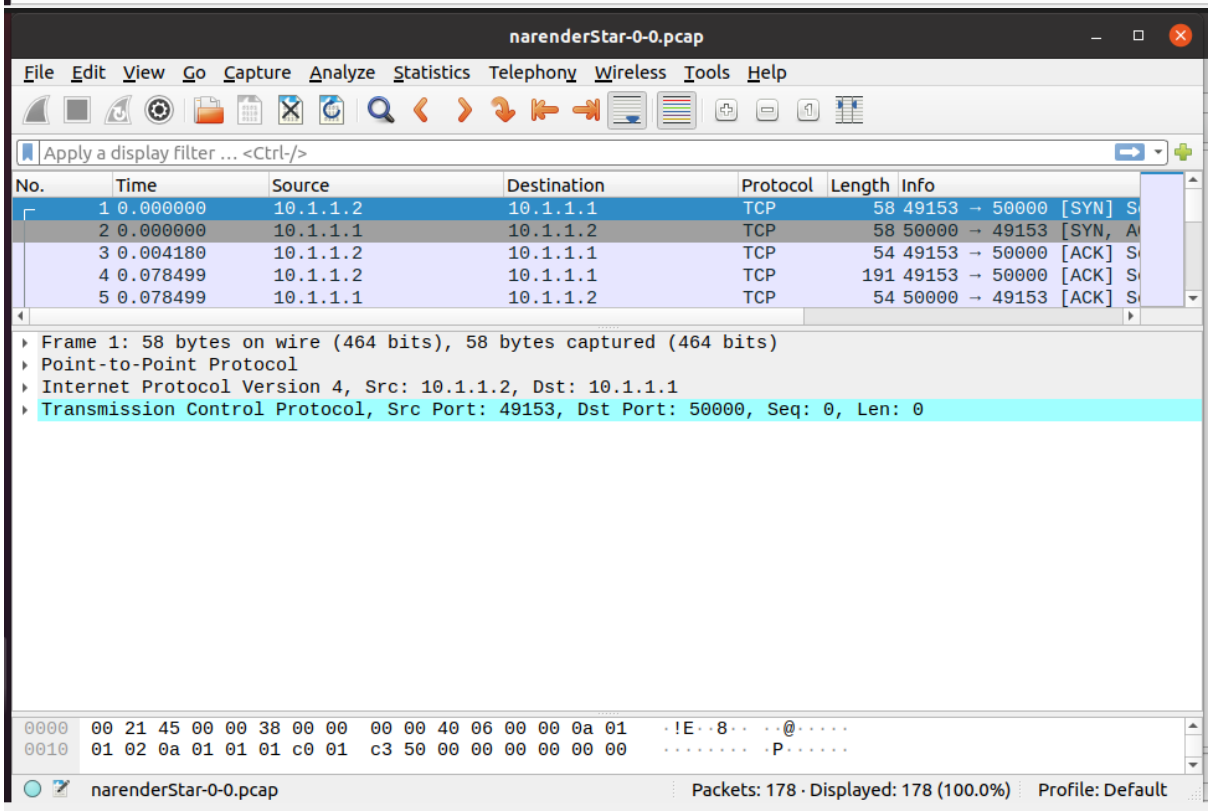
```
// Create a PointToPointStarHelper in order to easily create star topologies using p2p links.

 NS_LOG_INFO ("Install internet stack on all nodes.");
 InternetStackHelper internet;
 star.InstallStack (internet);

 NS_LOG_INFO ("Assign IP Addresses.");
 star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0", "255.255.255.0"));

 NS_LOG_INFO ("Create applications.");
 //
// Create a packet sink on the star "hub" to receive packets.
 //
 uint16_t port = 50000;
 Address hubLocalAddress (InetSocketAddress (Ipv4Address::GetAny (), port));
 PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory", hubLocalAddress);
 ApplicationContainer hubApp = packetSinkHelper.Install (star.GetHub ());
 hubApp.Start (Seconds (1.0));
 hubApp.Stop (Seconds (10.0));
// A network socket is a software structure within a network node of a computer network that
serves as an endpoint for sending and receiving data across the network.
// PacketSinkHelper -Receive and consume traffic generated to an IP address and port.
//

 // Create OnOff applications to send TCP to the hub, one on each spoke node.
 //
 OnOffHelper onOffHelper ("ns3::TcpSocketFactory", Address ());
 onOffHelper.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Constant=1]"));
 onOffHelper.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=0]"));

 ApplicationContainer spokeApps;

 for (uint32_t i = 0; i < star.SpokeCount (); ++i)
 {
 AddressValue remoteAddress (InetSocketAddress (star.GetHubIpv4Address (i), port));
 onOffHelper.SetAttribute ("Remote", remoteAddress);
 spokeApps.Add (onOffHelper.Install (star.GetSpokeNode (i)));
 }
 spokeApps.Start (Seconds (1.0));
 spokeApps.Stop (Seconds (10.0));

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

 NS_LOG_INFO ("Enable pcap tracing.");
//
 // Do pcap tracing on all point-to-point devices on all nodes.
 //
 pointToPoint.EnablePcapAll ("narenderStar");
```

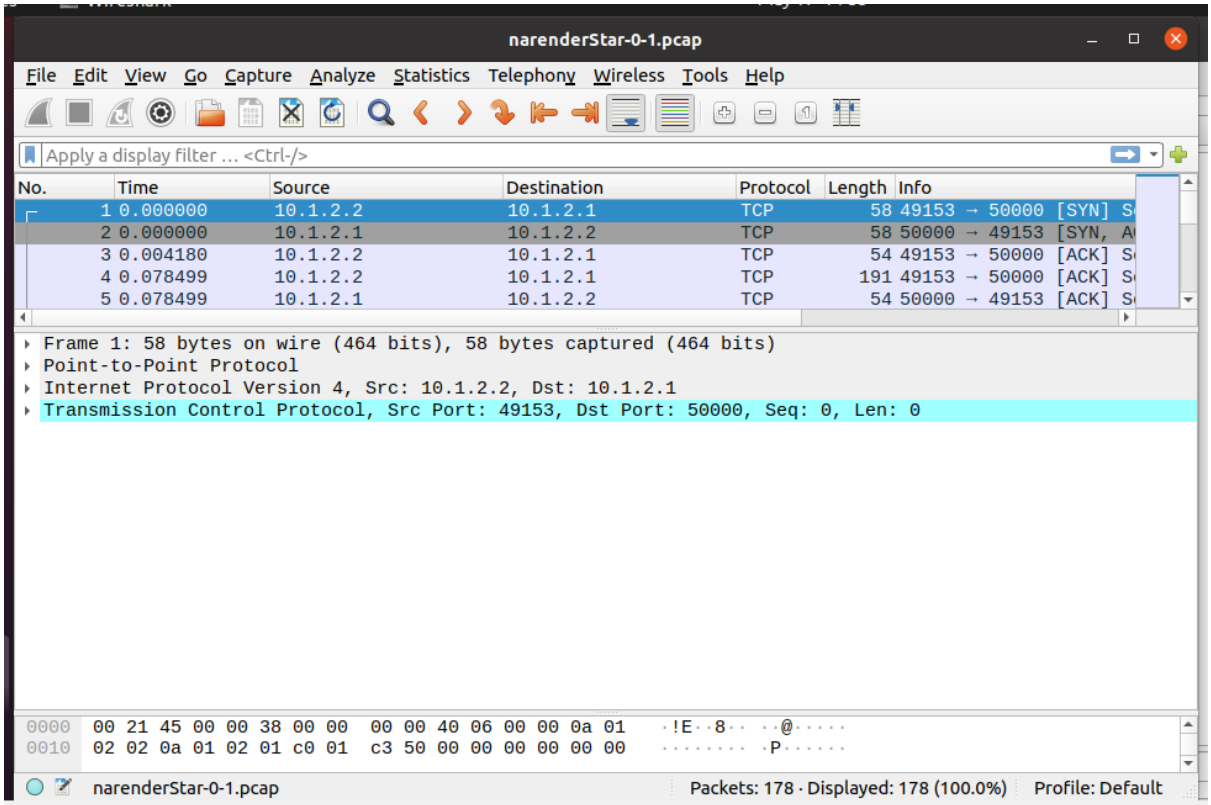
```
 NS_LOG_INFO ("Run Simulation.");

 star.BoundingBox(1,1,100,100);
 AnimationInterface anim("narenderStarAnim.xml");


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

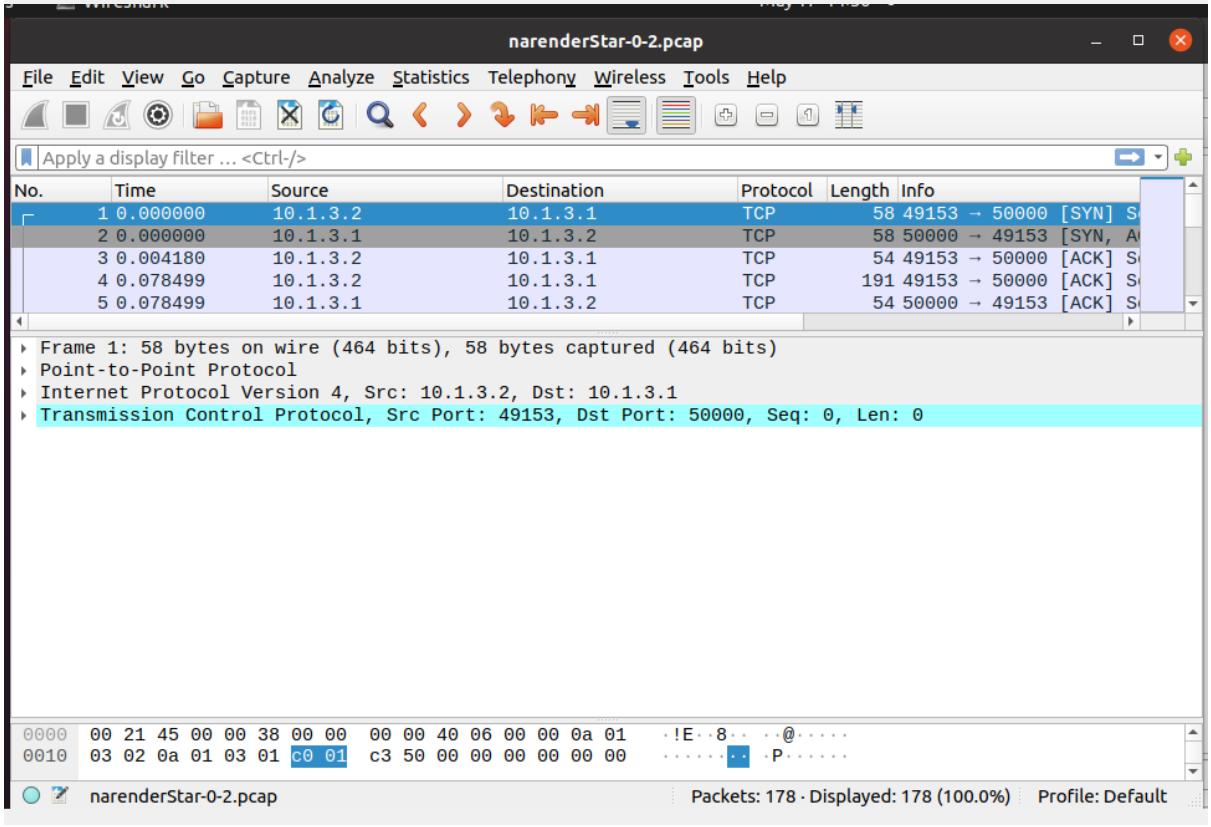
OUTPUT:

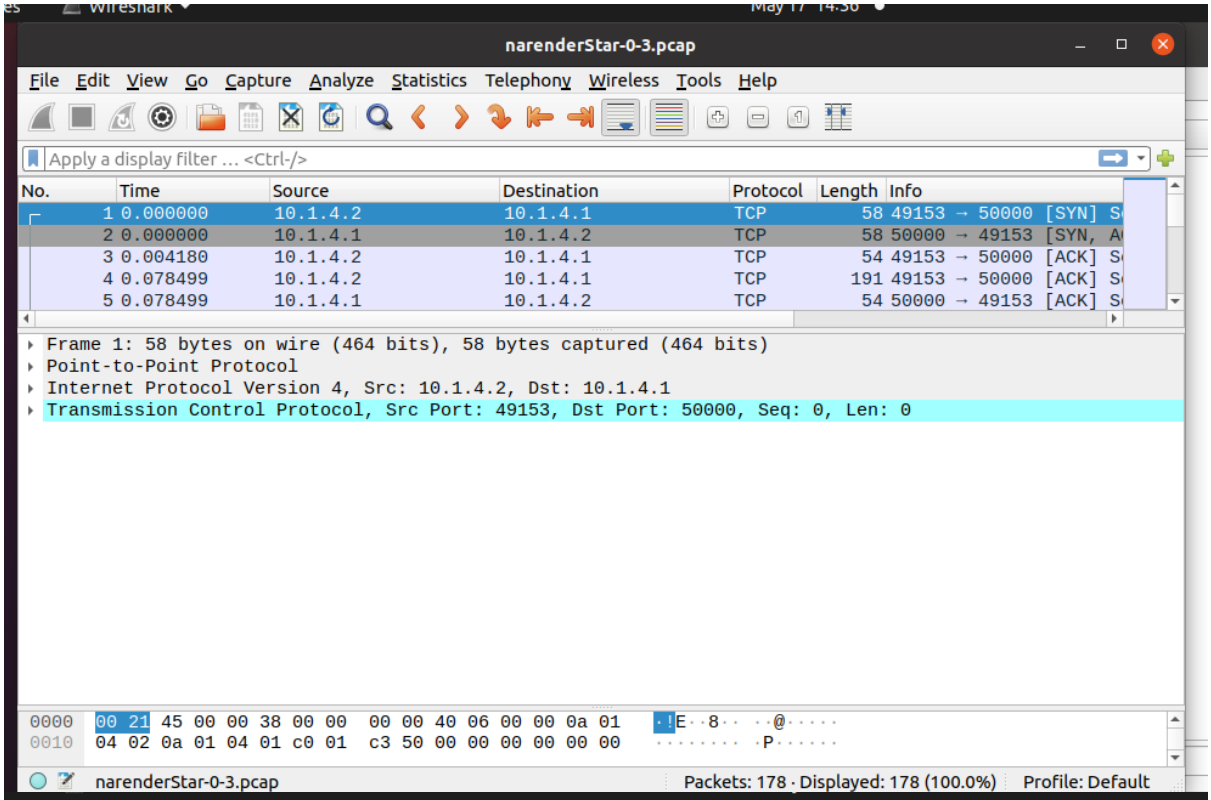


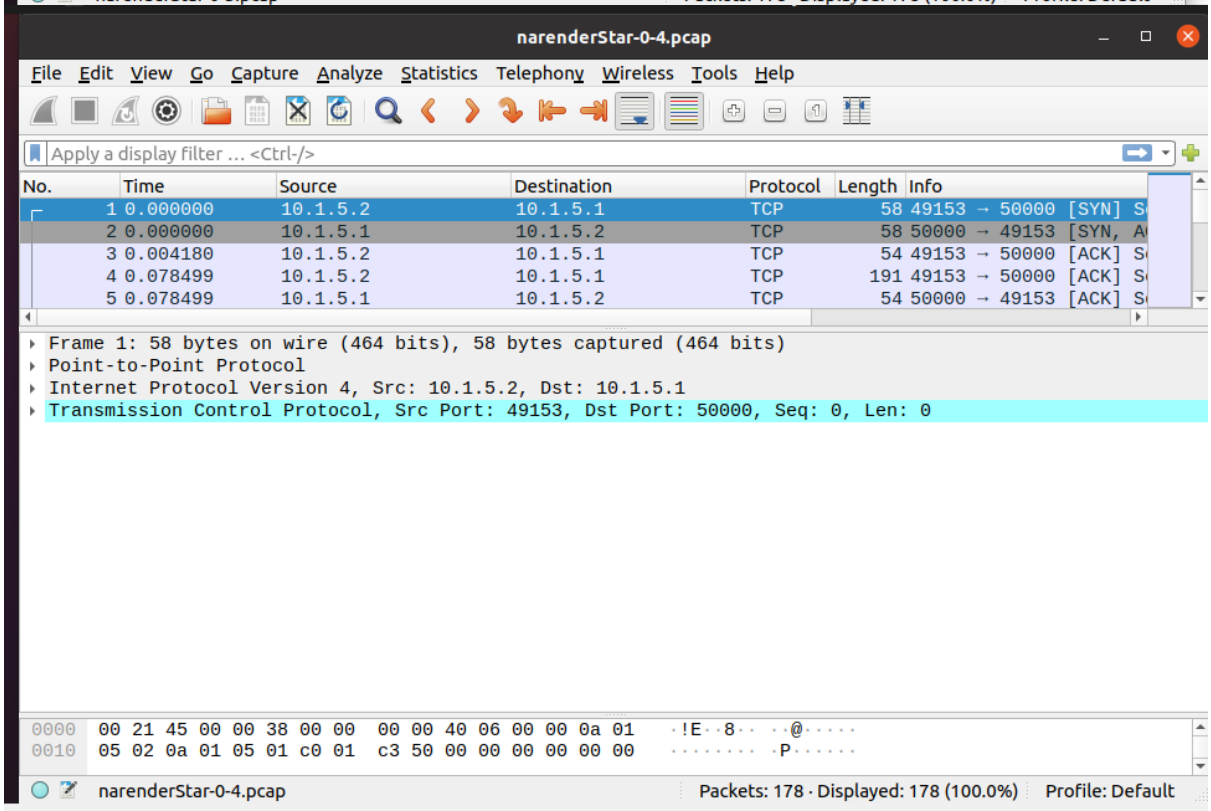

NetAnim
Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
Lines 5
Node Size 1
IP
MAC
T
0.0,0.0
37.6,0.0
75.3,0.0
0.0,37.6
37.6,37.6
75.3,37.6
0.0,75.3
37.6,75.3
75.3,75.3
0
1
2
3
4
5
6
7
8
Playing

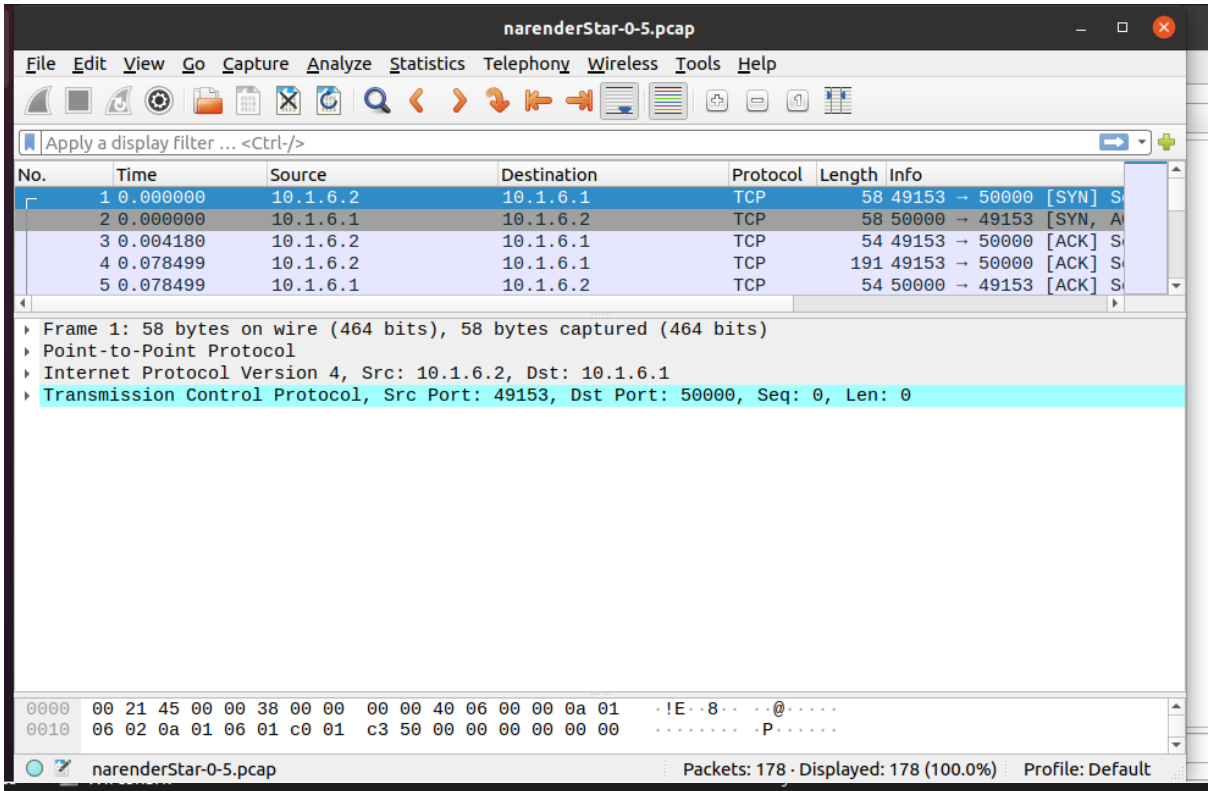

narenderStar-0-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.1.2 10.1.1.1 TCP 58 49153 → 50000 [SYN]
2 0.000000 10.1.1.1 10.1.1.2 TCP 58 50000 → 49153 [SYN, A
3 0.004180 10.1.1.2 10.1.1.1 TCP 54 49153 → 50000 [ACK]
4 0.078499 10.1.1.2 10.1.1.1 TCP 191 49153 → 50000 [ACK]
5 0.078499 10.1.1.1 10.1.1.2 TCP 54 50000 → 49153 [ACK]
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.1.2, Dst: 10.1.1.1
Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01
0010 01 02 0a 01 01 01 c0 01 c3 50 00 00 00 00 00 00
narenderStar-0-0.pcap
Packets: 178 · Displayed: 178 (100.0%)
Profile: Default

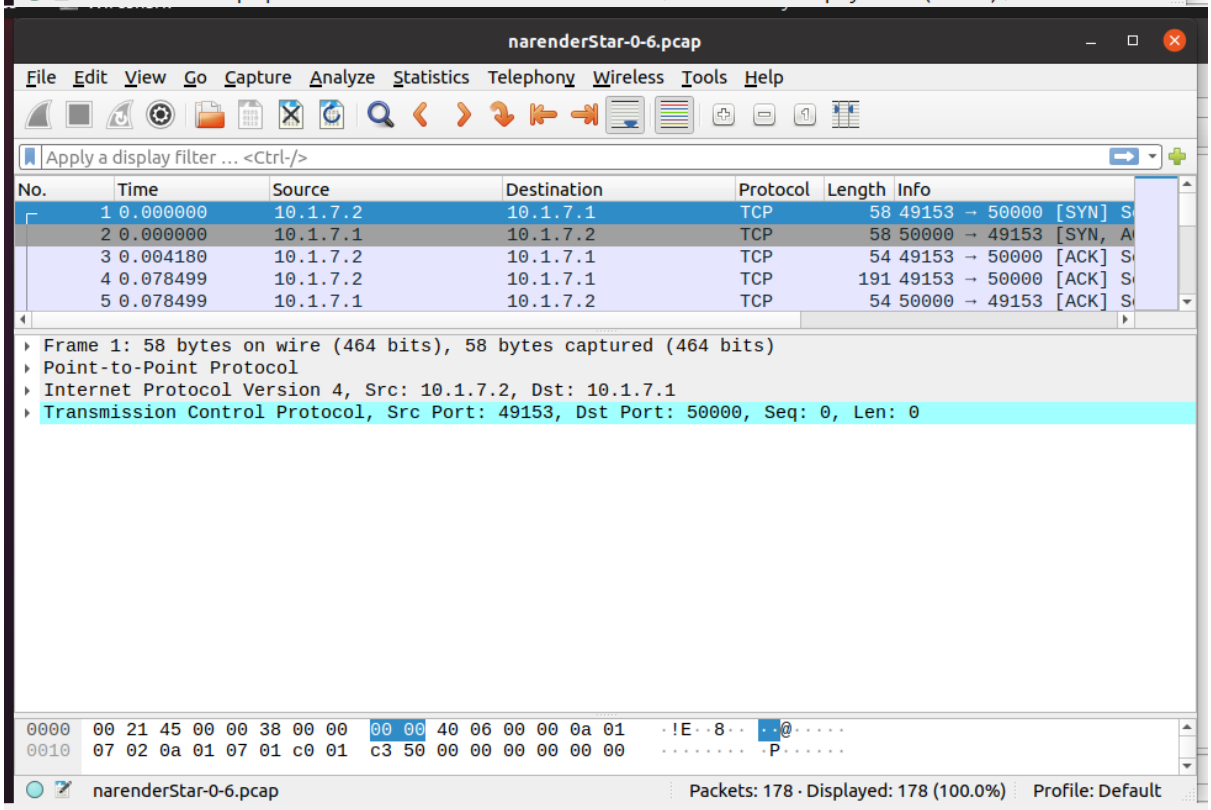

narenderStar-0-1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.2.2 10.1.2.1 TCP 58 49153 → 50000 [SYN]
2 0.000000 10.1.2.1 10.1.2.2 TCP 58 50000 → 49153 [SYN,
3 0.004180 10.1.2.2 10.1.2.1 TCP 54 49153 → 50000 [ACK]
4 0.078499 10.1.2.2 10.1.2.1 TCP 191 49153 → 50000 [ACK]
5 0.078499 10.1.2.1 10.1.2.2 TCP 54 50000 → 49153 [ACK]
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.2.2, Dst: 10.1.2.1
Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01
0010 02 02 0a 01 02 01 c0 01 c3 50 00 00 00 00 00 00
narenderStar-0-1.pcap
Packets: 178 · Displayed: 178 (100.0%)
Profile: Default

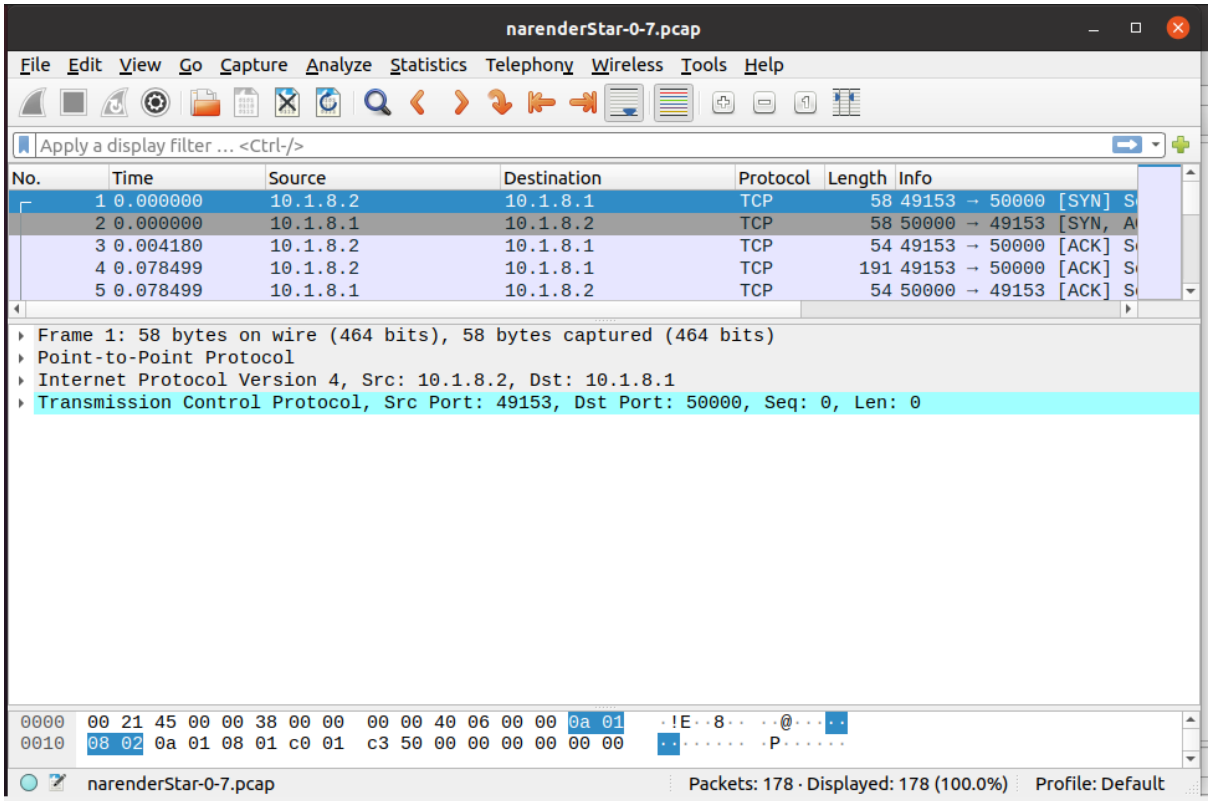

narenderStar-0-2.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.3.2 10.1.3.1 TCP 58 49153 → 50000 [SYN]
2 0.000000 10.1.3.1 10.1.3.2 TCP 58 50000 → 49153 [SYN,
3 0.004180 10.1.3.2 10.1.3.1 TCP 54 49153 → 50000 [ACK]
4 0.078499 10.1.3.2 10.1.3.1 TCP 191 49153 → 50000 [ACK]
5 0.078499 10.1.3.1 10.1.3.2 TCP 54 50000 → 49153 [ACK]
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.3.2, Dst: 10.1.3.1
Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01
0010 03 02 0a 01 03 01 c0 01 c3 50 00 00 00 00 00 00
narenderStar-0-2.pcap
Packets: 178 · Displayed: 178 (100.0%)
Profile: Default

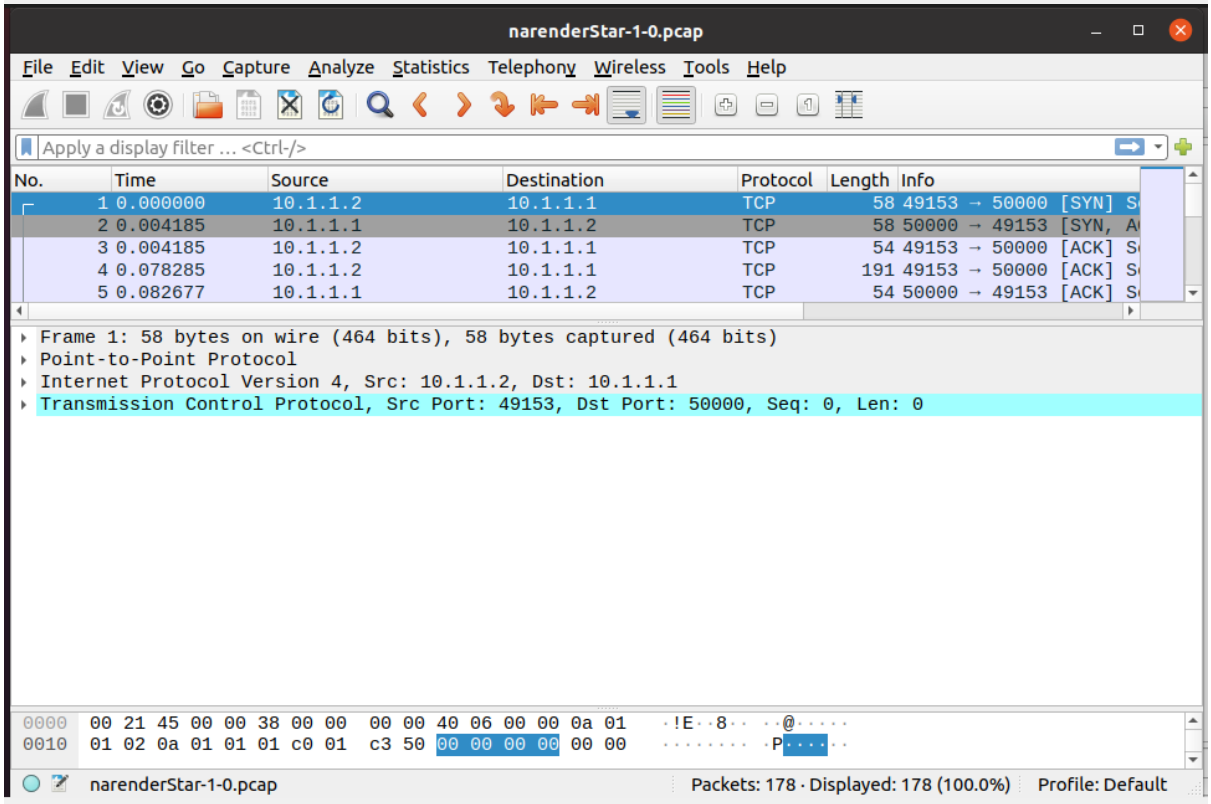

narenderStar-0-3.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.4.2 10.1.4.1 TCP 58 49153 → 50000 [SYN]
2 0.000000 10.1.4.1 10.1.4.2 TCP 58 50000 → 49153 [SYN, A
3 0.004180 10.1.4.2 10.1.4.1 TCP 54 49153 → 50000 [ACK]
4 0.078499 10.1.4.2 10.1.4.1 TCP 191 49153 → 50000 [ACK]
5 0.078499 10.1.4.1 10.1.4.2 TCP 54 50000 → 49153 [ACK]
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.4.2, Dst: 10.1.4.1
Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01
0010 04 02 0a 01 04 01 c0 01 c3 50 00 00 00 00 00 00
narenderStar-0-3.pcap
Packets: 178 · Displayed: 178 (100.0%)
Profile: Default

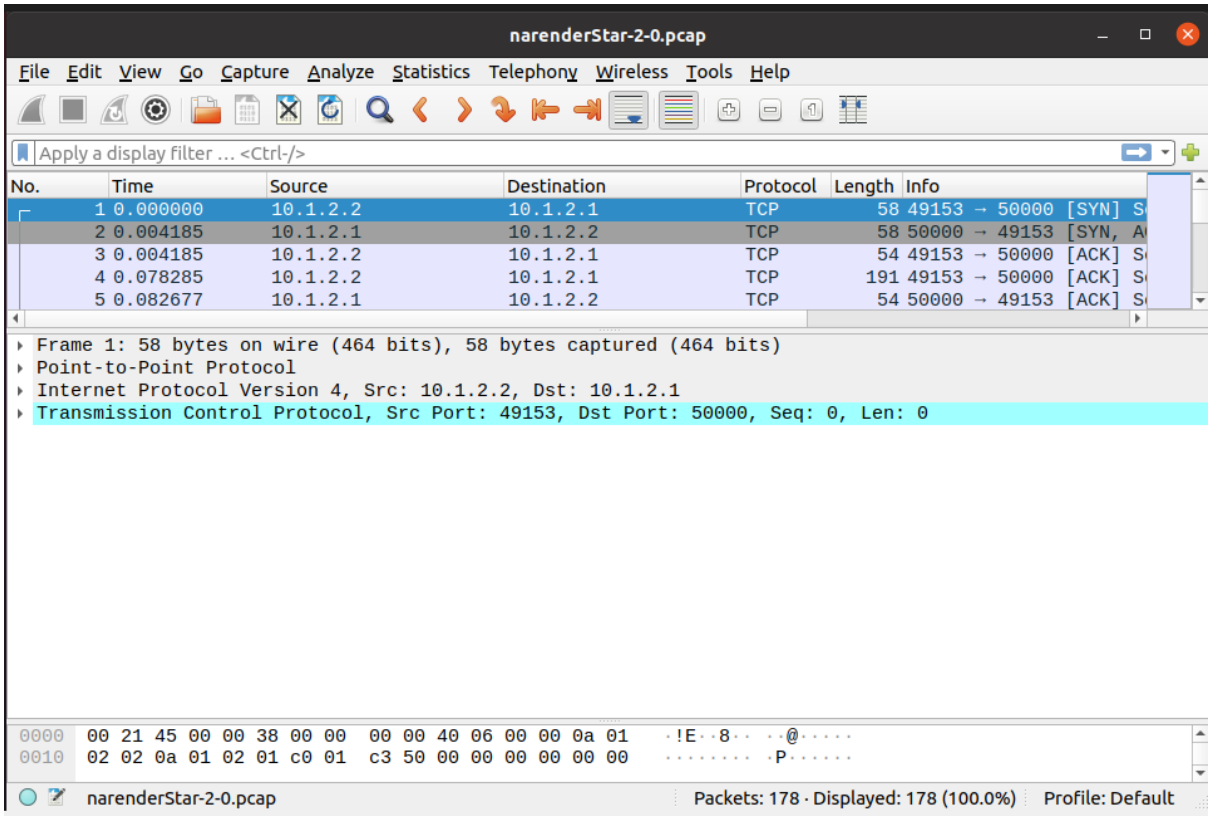

narenderStar-0-4.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.5.2 10.1.5.1 TCP 58 49153 → 50000 [SYN]
2 0.000000 10.1.5.1 10.1.5.2 TCP 58 50000 → 49153 [SYN, A
3 0.004180 10.1.5.2 10.1.5.1 TCP 54 49153 → 50000 [ACK]
4 0.078499 10.1.5.2 10.1.5.1 TCP 191 49153 → 50000 [ACK]
5 0.078499 10.1.5.1 10.1.5.2 TCP 54 50000 → 49153 [ACK]
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.5.2, Dst: 10.1.5.1
Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01
0010 05 02 0a 01 05 01 c0 01 c3 50 00 00 00 00 00 00
narenderStar-0-4.pcap
Packets: 178 · Displayed: 178 (100.0%)
Profile: Default

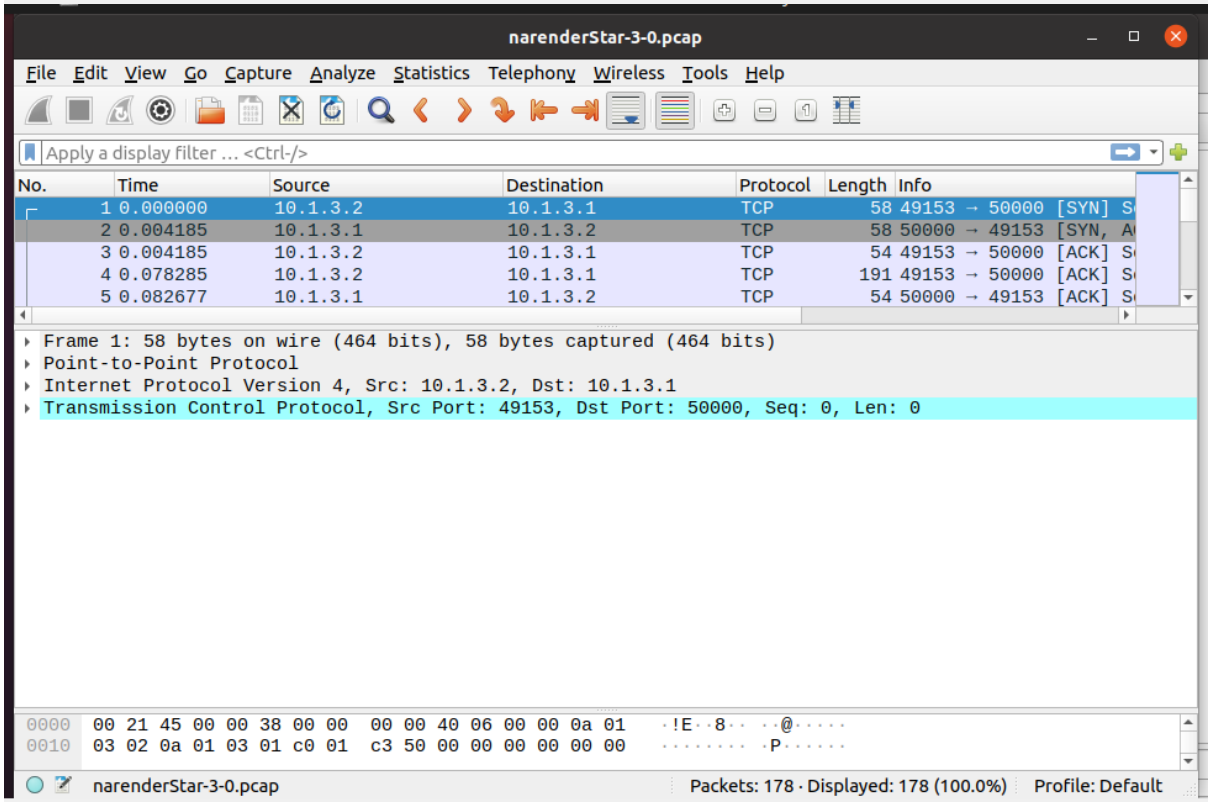

narenderStar-0-5.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.6.2 10.1.6.1 TCP 58 49153 → 50000 [SYN]
2 0.000000 10.1.6.1 10.1.6.2 TCP 58 50000 → 49153 [SYN, A
3 0.004180 10.1.6.2 10.1.6.1 TCP 54 49153 → 50000 [ACK]
4 0.078499 10.1.6.2 10.1.6.1 TCP 191 49153 → 50000 [ACK]
5 0.078499 10.1.6.1 10.1.6.2 TCP 54 50000 → 49153 [ACK]
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.6.2, Dst: 10.1.6.1
Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01
0010 06 02 0a 01 06 01 c0 01 c3 50 00 00 00 00 00 00
narenderStar-0-5.pcap
Packets: 178 · Displayed: 178 (100.0%)
Profile: Default

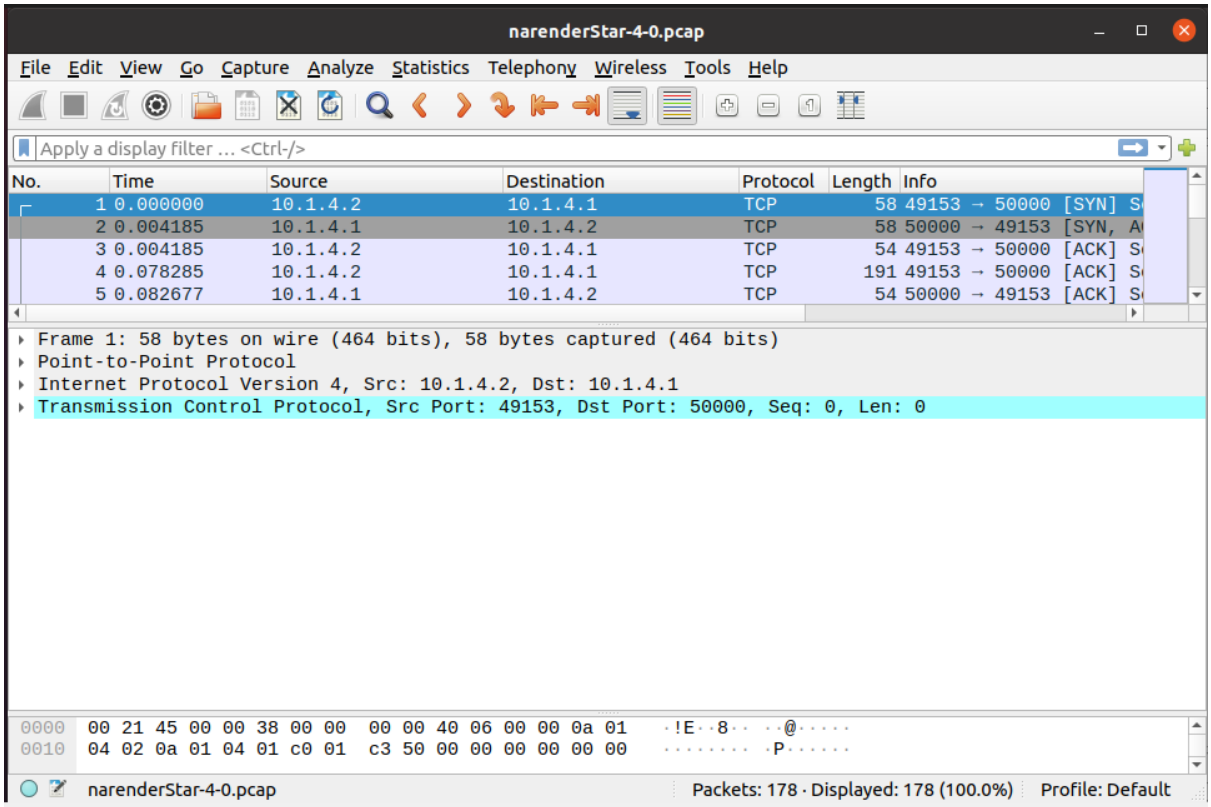

narenderStar-0-6.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.7.2 10.1.7.1 TCP 58 49153 → 50000 [SYN]
2 0.000000 10.1.7.1 10.1.7.2 TCP 58 50000 → 49153 [SYN, A
3 0.004180 10.1.7.2 10.1.7.1 TCP 54 49153 → 50000 [ACK]
4 0.078499 10.1.7.2 10.1.7.1 TCP 191 49153 → 50000 [ACK]
5 0.078499 10.1.7.1 10.1.7.2 TCP 54 50000 → 49153 [ACK]
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.7.2, Dst: 10.1.7.1
Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01
0010 07 02 0a 01 07 01 c0 01 c3 50 00 00 00 00 00 00
narenderStar-0-6.pcap
Packets: 178 · Displayed: 178 (100.0%)
Profile: Default

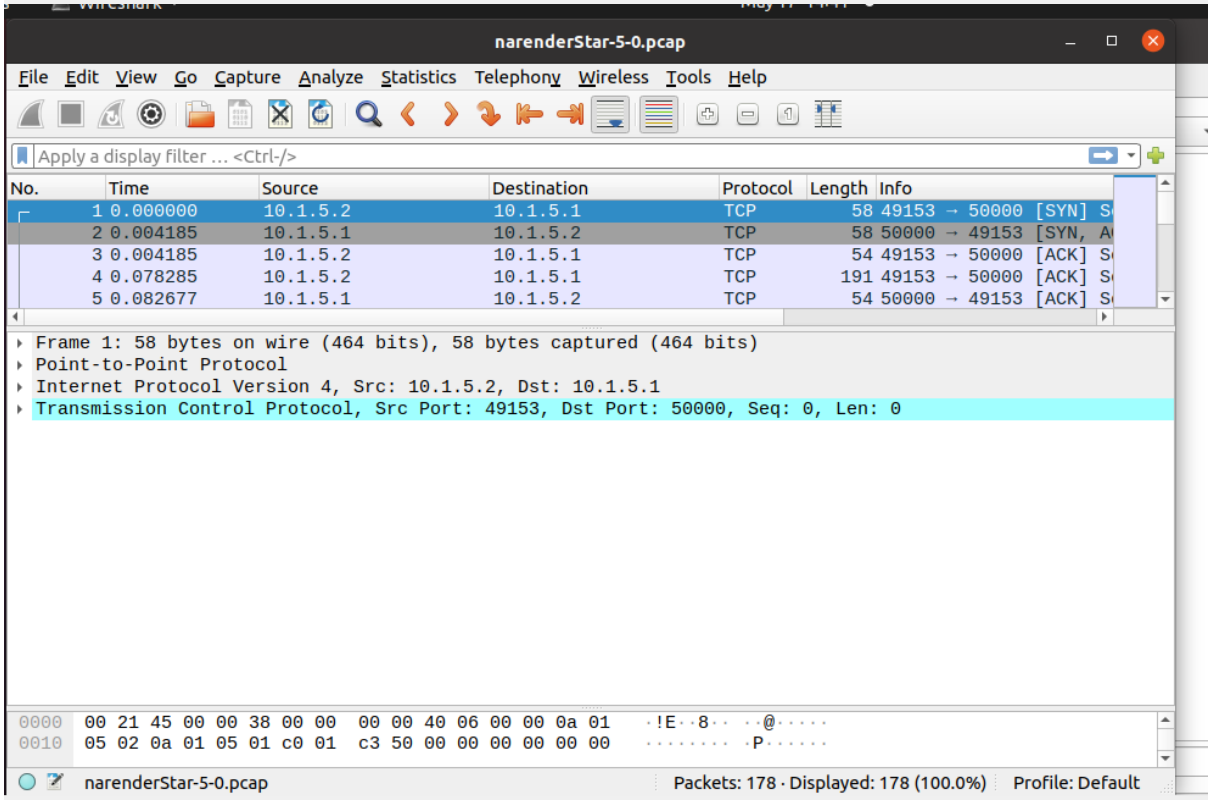

narenderStar-0-7.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.8.2 10.1.8.1 TCP 58 49153 → 50000 [SYN]
2 0.000000 10.1.8.1 10.1.8.2 TCP 58 50000 → 49153 [SYN,
3 0.004180 10.1.8.2 10.1.8.1 TCP 54 49153 → 50000 [ACK]
4 0.078499 10.1.8.2 10.1.8.1 TCP 191 49153 → 50000 [ACK]
5 0.078499 10.1.8.1 10.1.8.2 TCP 54 50000 → 49153 [ACK]
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.8.2, Dst: 10.1.8.1
Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01
0010 08 02 0a 01 08 01 c0 01 c3 50 00 00 00 00 00 00
narenderStar-0-7.pcap
Packets: 178 · Displayed: 178 (100.0%)
Profile: Default

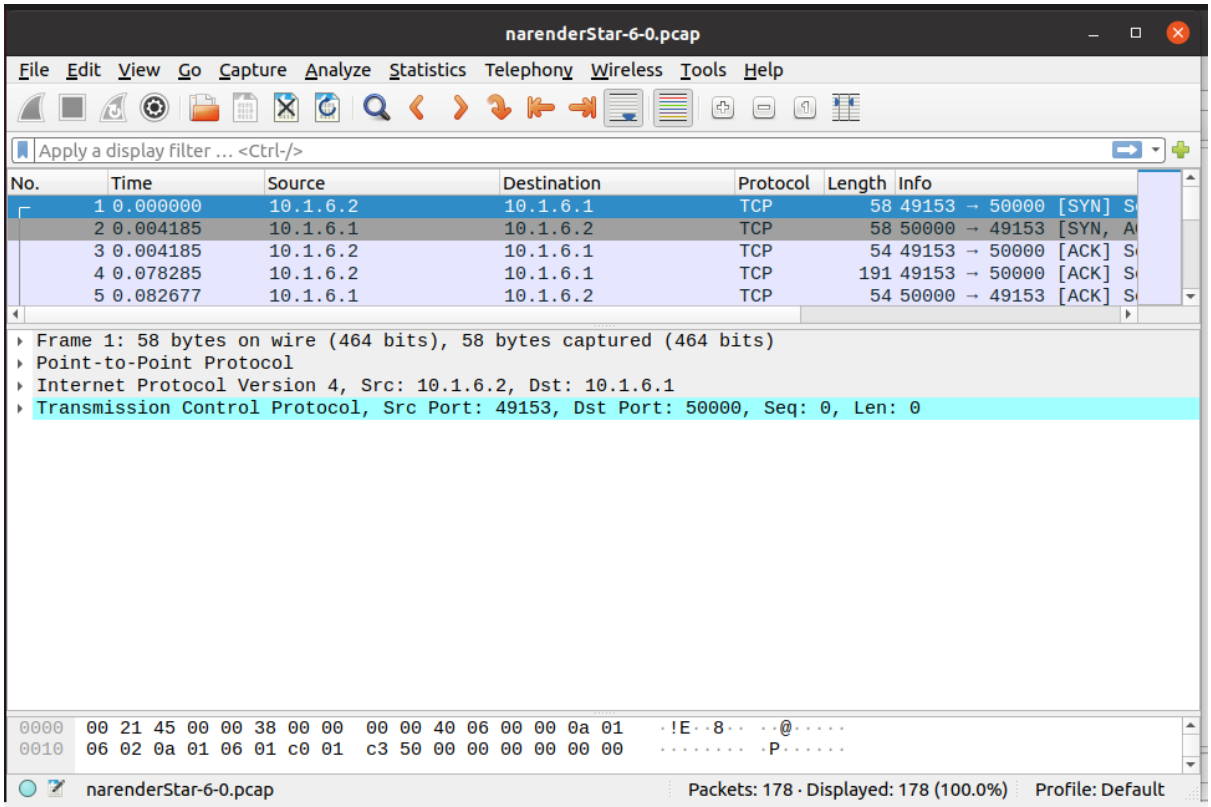

narenderStar-1-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.1.2 10.1.1.1 TCP 58 49153 → 50000 [SYN]
2 0.004185 10.1.1.1 10.1.1.2 TCP 58 50000 → 49153 [SYN,
3 0.004185 10.1.1.2 10.1.1.1 TCP 54 49153 → 50000 [ACK]
4 0.078285 10.1.1.2 10.1.1.1 TCP 191 49153 → 50000 [ACK]
5 0.082677 10.1.1.1 10.1.1.2 TCP 54 50000 → 49153 [ACK]
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.1.2, Dst: 10.1.1.1
Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01
0010 01 02 0a 01 01 01 c0 01 c3 50 00 00 00 00 00 00
narenderStar-1-0.pcap
Packets: 178 · Displayed: 178 (100.0%)
Profile: Default

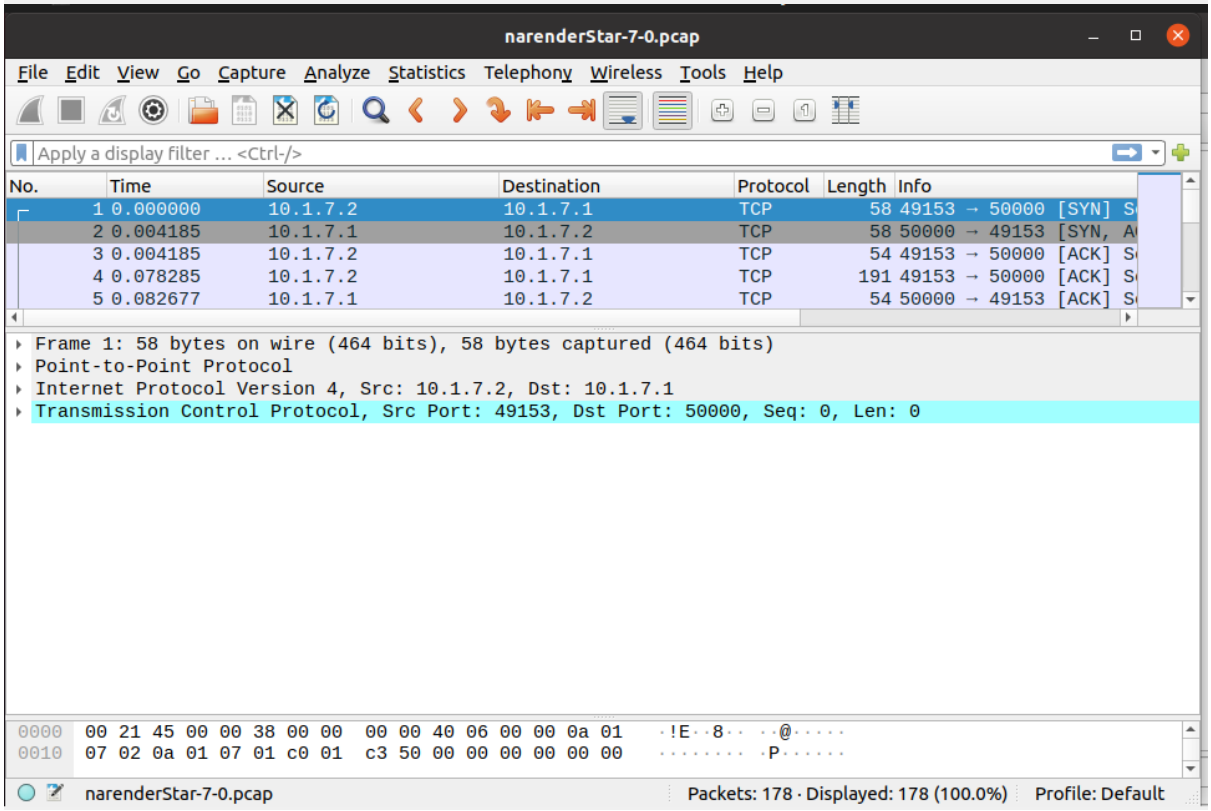

narenderStar-2-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.2.2 10.1.2.1 TCP 58 49153 → 50000 [SYN]
2 0.004185 10.1.2.1 10.1.2.2 TCP 58 50000 → 49153 [SYN,
3 0.004185 10.1.2.2 10.1.2.1 TCP 54 49153 → 50000 [ACK]
4 0.078285 10.1.2.2 10.1.2.1 TCP 191 49153 → 50000 [ACK]
5 0.082677 10.1.2.1 10.1.2.2 TCP 54 50000 → 49153 [ACK]
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.2.2, Dst: 10.1.2.1
Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01
0010 02 02 0a 01 02 01 c0 01 c3 50 00 00 00 00 00 00
narenderStar-2-0.pcap
Packets: 178 · Displayed: 178 (100.0%)
Profile: Default


narenderStar-3-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.3.2 10.1.3.1 TCP 58 49153 → 50000 [SYN]
2 0.004185 10.1.3.1 10.1.3.2 TCP 58 50000 → 49153 [SYN,
3 0.004185 10.1.3.2 10.1.3.1 TCP 54 49153 → 50000 [ACK]
4 0.078285 10.1.3.2 10.1.3.1 TCP 191 49153 → 50000 [ACK]
5 0.082677 10.1.3.1 10.1.3.2 TCP 54 50000 → 49153 [ACK]
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.3.2, Dst: 10.1.3.1
Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01
0010 03 02 0a 01 03 01 c0 01 c3 50 00 00 00 00 00 00
narenderStar-3-0.pcap
Packets: 178 · Displayed: 178 (100.0%)
Profile: Default


narenderStar-4-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.4.2 10.1.4.1 TCP 58 49153 → 50000 [SYN] S
2 0.004185 10.1.4.1 10.1.4.2 TCP 58 50000 → 49153 [SYN, A
3 0.004185 10.1.4.2 10.1.4.1 TCP 54 49153 → 50000 [ACK] S
4 0.078285 10.1.4.2 10.1.4.1 TCP 191 49153 → 50000 [ACK] S
5 0.082677 10.1.4.1 10.1.4.2 TCP 54 50000 → 49153 [ACK] S
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.4.2, Dst: 10.1.4.1
Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01 !E 8 @
0010 04 02 0a 01 04 01 c0 01 c3 50 00 00 00 00 00 00 P
narenderStar-4-0.pcap
Packets: 178 · Displayed: 178 (100.0%)
Profile: Default


narenderStar-5-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 10.1.5.2 10.1.5.1 TCP 58 49153 → 50000 [SYN] S
2 0.004185 10.1.5.1 10.1.5.2 TCP 58 50000 → 49153 [SYN, A
3 0.004185 10.1.5.2 10.1.5.1 TCP 54 49153 → 50000 [ACK] S
4 0.078285 10.1.5.2 10.1.5.1 TCP 191 49153 → 50000 [ACK] S
5 0.082677 10.1.5.1 10.1.5.2 TCP 54 50000 → 49153 [ACK] S
Frame 1: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
Point-to-Point Protocol
Internet Protocol Version 4, Src: 10.1.5.2, Dst: 10.1.5.1
Transmission Control Protocol, Src Port: 49153, Dst Port: 50000, Seq: 0, Len: 0
0000 00 21 45 00 00 38 00 00 00 00 40 06 00 00 0a 01 !E 8 @
0010 05 02 0a 01 05 01 c0 01 c3 50 00 00 00 00 00 00 P
narenderStar-5-0.pcap
Packets: 178 · Displayed: 178 (100.0%)
Profile: Default





CONCLUSION:

From this practical, I have learned about star topology simulation in ns3.

Aim: Program to simulate Bus topology.

THEORY:

Understanding Network Topology:

Network topology is the arrangement of the elements (links, nodes, etc.) of a communication network. Network topology can be used to define or describe the arrangement of various types of telecommunication networks, including command and control radio networks, industrial field busses and computer networks.

Network topology is the topological structure of a network and may be depicted physically or logically. It is an application of graph theory wherein communicating devices are modelled as nodes and the connections between the devices are modelled as links or lines between the nodes.

Physical topology is the placement of the various components of a network (e.g., device location and cable installation), while logical topology illustrates how data flows within a network. Distances between nodes, physical interconnections, transmission rates, or signal types may differ between two different networks, yet their logical topologies may be identical.

A network's physical topology is a particular concern of the physical layer of the OSI model.

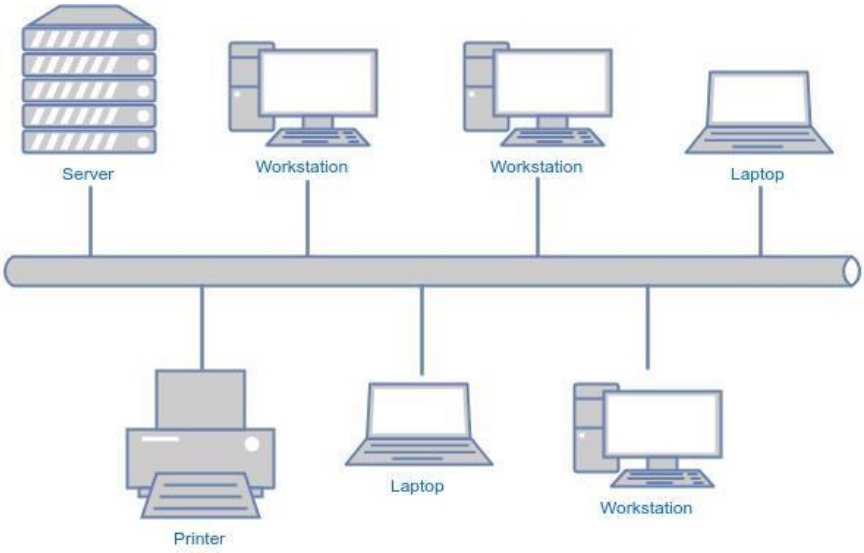
Examples of network topologies are found in local area networks (LAN), a common computer network installation. Any given node in the LAN has one or more physical links to other devices in the network; graphically mapping these links results in a geometric shape that can be used to describe the physical topology of the network.

Bus Topology:

Bus topology is a specific kind of network topology in which all of the various devices in the network are connected to a single cable or line. In general, the term refers to how various devices are set up in a network.

Alternatively mentioned as line topology, bus topology could even be a specific quite topology during which each computer and network device is connected to a minimum of one cable or backbone. In general, term refers to how various devices are acknowledged during a network. counting on sort of network card, a coax or an RJ-45 network cable is employed to attach them together.

Bus topology carries transmitted data through cable. because data reaches each node, node checks destination address (MAC/IP address) to work out if it matches their address. If address does not match with node, node does nothing more. But if addresses of node match to address contained within data, then they process on knowledge. In bus, communication between nodes are done through foremost network cable.



Bus Topology Network

Advantages of Bus Topology:

- It is the easiest network topology for connecting peripherals or computers in a linear fashion.
- It works very efficient well when there is a small network.
- Length of cable required is less than a star topology.
- It is easy to connect or remove devices in this network without affecting any other device. • Very cost-effective as compared to other network topology i.e. mesh and star
- It is easy to understand topology.
- Easy to expand by joining the two cables together.

Disadvantages of Bus Topology:

- Bus topology is not great for large networks.
- Identification of problem becomes difficult if whole network goes down.
- Troubleshooting of individual device issues is very hard.
- Need of terminators are required at both ends of main cable.
- Additional devices slow network down.
- If a main cable is damaged, whole network fails or splits into two.
- Packet loss is high.
- This network topology is very slow as compared to other topologies.

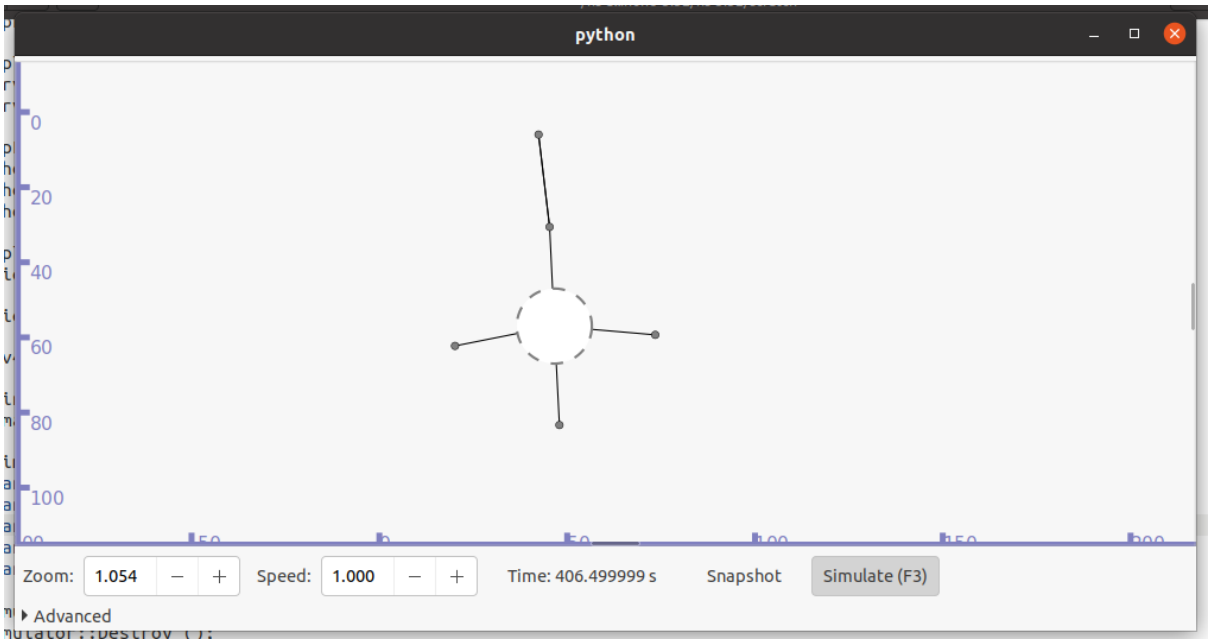
SOURCE CODE:

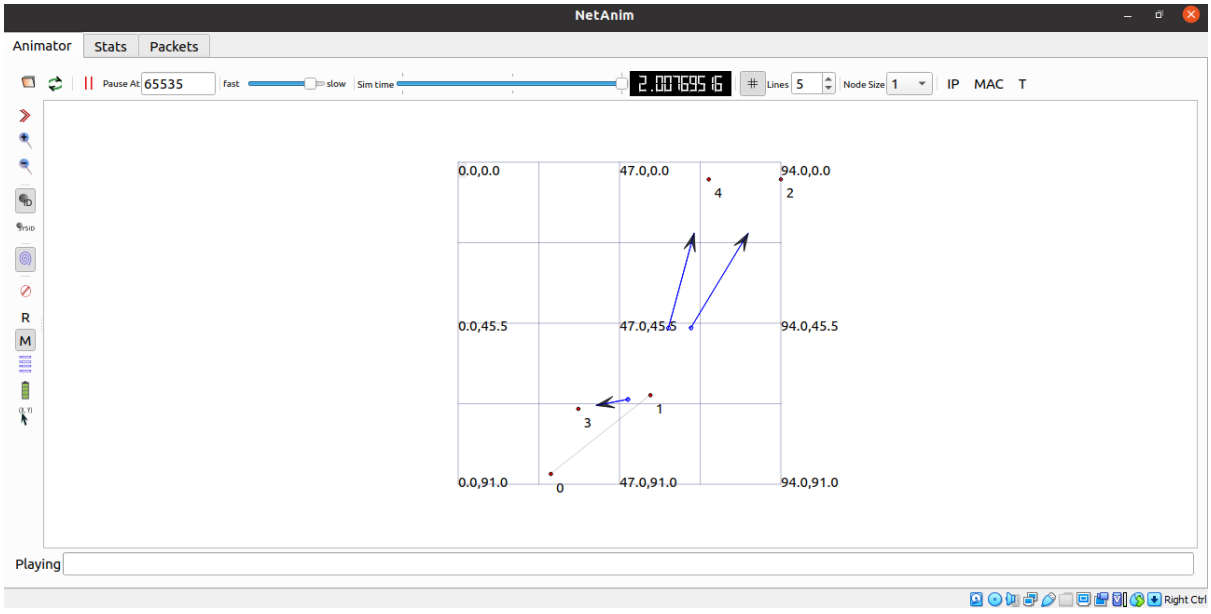
```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/netanim-module.h"

// Default Network Topology
//
//       172.16.1.0
// n0 ---------- n1   n2   n3   n4
//    point-to-point  |    |    |    |
//                    ================
//                      LAN 172.16.2.0


using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("SecondScriptExample");

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

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

  cmd.Parse (argc,argv);

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

  nCsma = nCsma == 0 ? 1 : nCsma;

  NodeContainer p2pNodes;
  p2pNodes.Create (2);

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

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

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

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

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

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

Ipv4AddressHelper address;
address.SetBase ("172.16.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign (p2pDevices);

address.SetBase ("172.16.2.0", "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 ("second2");
csma.EnablePcap ("second2", csmaDevices.Get (2), true);

AnimationInterface anim ("anim3.xml");
//anim.SetConstantPosition (p2pNodes.Get(1),10.0,10.0);
//anim.SetConstantPosition (p2pNodes.Get(2),20.0,20.0);
//anim.SetConstantPosition (csmaNodes.Get(1),10.0,10.0);
//anim.SetConstantPosition (csmaNodes.Get(2),10.0,10.0);
//anim.SetConstantPosition (csmaNodes.Get(3),10.0,10.0);
```

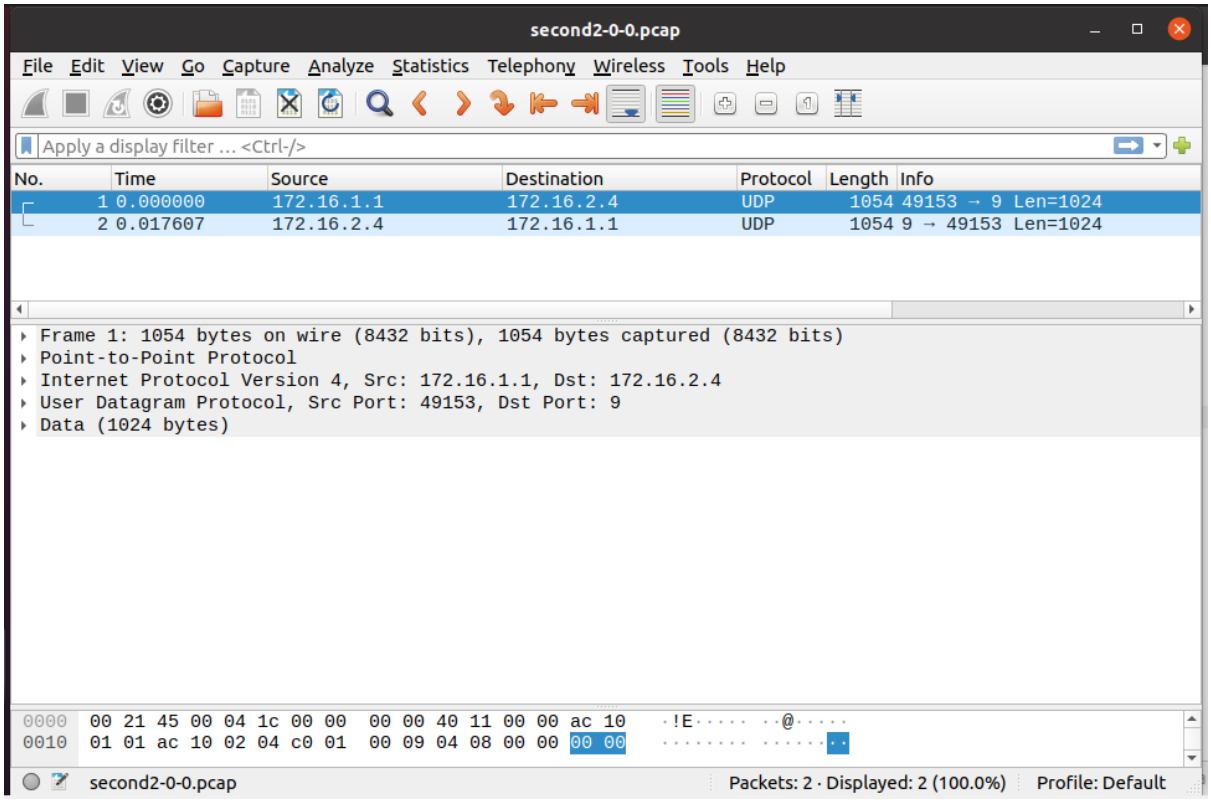
```
  Simulator::Run ();
  Simulator::Destroy ();
  return 0;
}
```

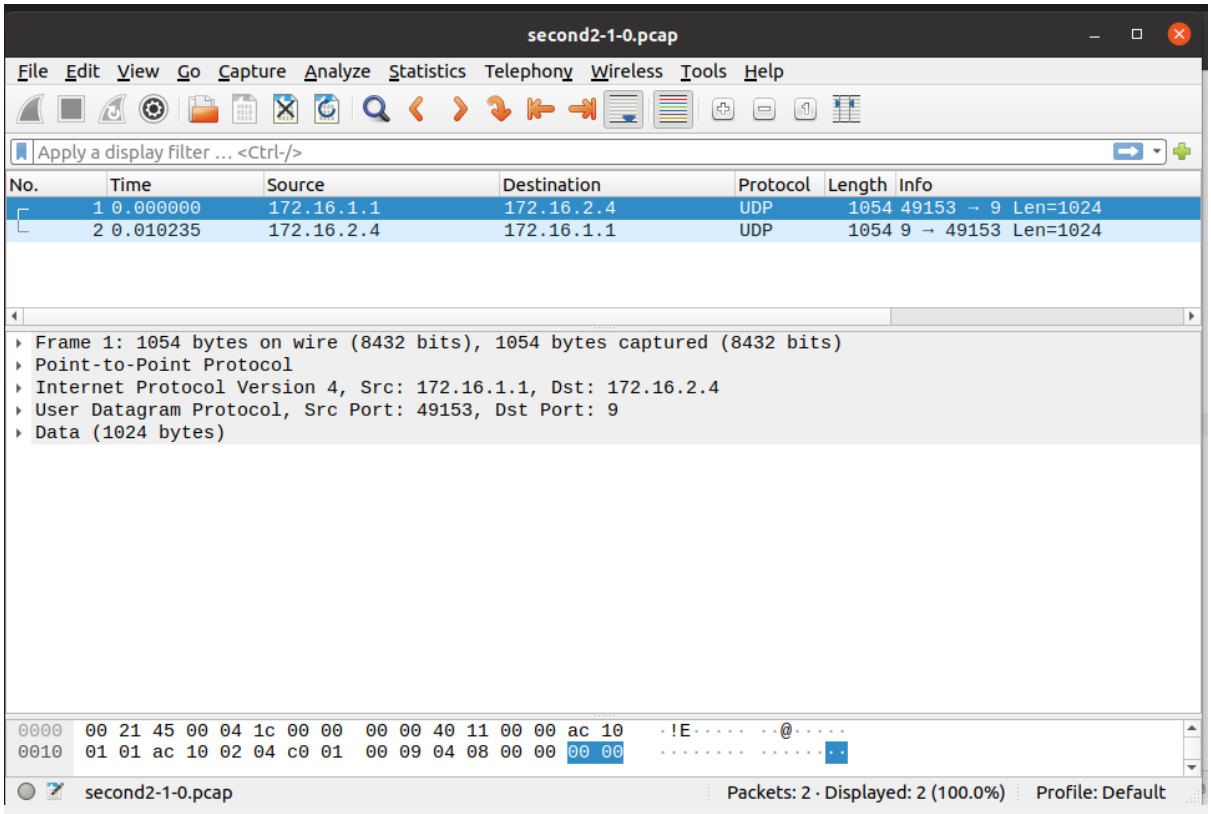
OUTPUT:

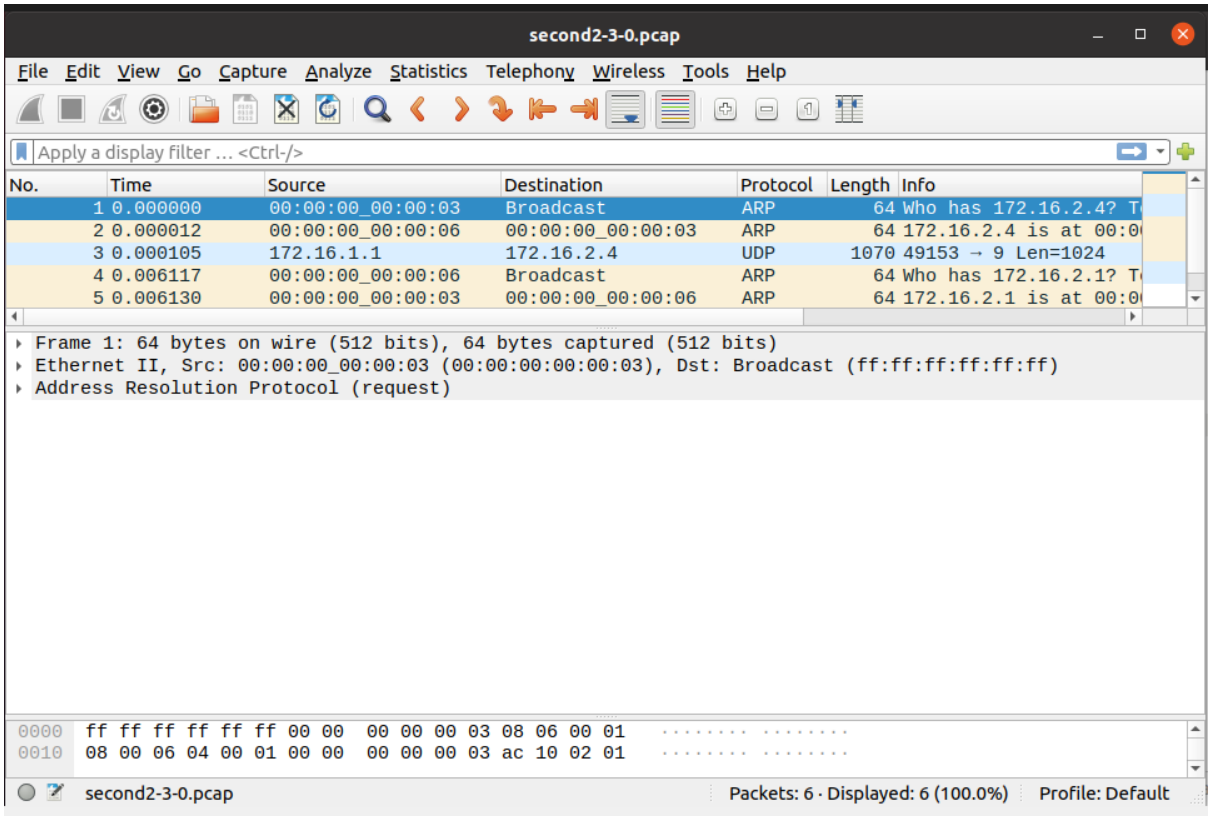





```
vaish@vaish-VirtualBox:~/ns-allinone-3.32/ns-3.32$ wireshark second2-0-0.pcap
^C
vaish@vaish-VirtualBox:~/ns-allinone-3.32/ns-3.32$ wireshark second2-1-0.pcap
^C
vaish@vaish-VirtualBox:~/ns-allinone-3.32/ns-3.32$ wireshark second2-3-0.pcap
vaish@vaish-VirtualBox:~/ns-allinone-3.32/ns-3.32$
```


second2-0-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 172.16.1.1 172.16.2.4 UDP 1054 49153 → 9 Len=1024
2 0.017607 172.16.2.4 172.16.1.1 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: 172.16.1.1, Dst: 172.16.2.4
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 ac 10
0010 01 01 ac 10 02 04 c0 01 00 09 04 08 00 00 00 00
second2-0-0.pcap
Packets: 2 · Displayed: 2 (100.0%)
Profile: Default


second2-1-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 172.16.1.1 172.16.2.4 UDP 1054 49153 → 9 Len=1024
2 0.010235 172.16.2.4 172.16.1.1 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: 172.16.1.1, Dst: 172.16.2.4
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 ac 10
0010 01 01 ac 10 02 04 c0 01 00 09 04 08 00 00 00 00
second2-1-0.pcap
Packets: 2 · Displayed: 2 (100.0%)
Profile: Default



CONCLUSION:

From this practical, I have learned about bus topology simulation in ns3.

AIM: SIMULATION OF WIRELESS NETWORK IN NS3

THEORY:

What Is a Wireless Network or Wi-Fi?

A wireless network refers to a computer network that makes use of Radio Frequency (RF) connections between nodes in the network. Wireless networks are a popular solution for homes, businesses, and telecommunications networks.

It is common for people to wonder "what is a wireless network" because while they exist nearly everywhere people live and work, how they work is often a mystery. Similarly, people often assume that all wireless is Wi-Fi, and many would be surprised to discover that the two are not synonymous. Both use RF, but there are many different types of wireless networks across a range of technologies (Bluetooth, ZigBee, LTE, 5G), while Wi-Fi is specific to the wireless protocol defined by the Institute of Electrical and Electronic Engineers (IEEE) in the 802.11 specification and it's amendments.

Wired vs. Wireless Network: What Is the Difference?

At the most obvious, a wireless network keeps devices connected to a network while still allowing them the freedom to move about, unencumbered by wires. A wired network, on the other hand, makes use of cables that connect devices to the network. These devices are often desktop or laptop computers but can also include scanners and point-of-sale machines.

There are more subtle technology differences that come in to play between wired and wireless. Most modern wired networks are now "full duplex", meaning that they can be transmitting/receiving packets in both directions simultaneously. In addition, most wired networks have a dedicated cable that runs to each end user device.

In a Wi-Fi network, the medium (the radio frequency being used for the network) is a shared resource, not just for the users of the network, but often for other technologies as well (Wi-Fi operates in what are called 'shared' bands, where many different electronic devices are approved to operate). This has several implications: 1) unlike a wired network, wireless can't both talk and listen at the same time, it is "half duplex" 2) All users are sharing the same space must take turns to talk 3) everyone can 'hear' all traffic going on. This has forced Wi-Fi networks to implement various security measures over the years to protect the confidentiality of information passed wirelessly.

Examples of wireless networks:

- Mobile phone networks
- Wireless sensor networks
- Satellite communication networks
- Terrestrial microwave networks

Types of Wireless Connections

In addition to a LAN, there are a few other types of common wireless networks: personal-area network (PAN), metropolitan-area network (MAN), and wide-area network (WAN).

LAN

A local-area network is a computer network that exists at a single site, such as an office building. It can be used to connect a variety of components, such as computers, printers, and data storage devices. LANs consist of components like switches, access points, routers, firewalls, and Ethernet cables to tie it all together. Wi-Fi is the most commonly known wireless LAN.

PAN

A personal-area network consists of a network centralized around the devices of a single person in a single location. A PAN could have computers, phones, video game consoles, or other peripheral devices. They are common inside homes and small office buildings. Bluetooth is the most commonly known wireless PAN.

MAN

A metropolitan-area network is a computer network that spans across a city, small geographical area, or business or college campus. One feature that differentiates a MAN from a LAN is its size. A LAN usually consists of a solitary building or area. A MAN can cover several square miles, depending on the needs of the organization.

Large companies, for example, may use a MAN if they have a spacious campus and need to manage key components, such as HVAC and electrical systems.

WAN

A wide-area network covers a very large area, like an entire city, state, or country. In fact, the internet is a WAN. Like the internet, a WAN can contain smaller networks, including LANs or MANs. Cellular services are the most commonly known wireless WANs.

The Components of a Wireless Network

Several components make up a wireless network's topology:

1) **Clients:** What we tend to think of as the end user devices are typically called 'clients'. As the reach of Wi-Fi has expanded, a variety of devices may be using Wi-Fi to connect the network, including phones, tablets, laptops, desktops, and more. This gives users the ability to move about the area without sacrificing their bridge to the network. In some instances, mobility within an office, warehouse, or other work area is necessary. For example, if employees have to use scanners to register packages due to be shipped, a wireless network provides the flexibility they need to freely move about the warehouse.
2) **Access Point (AP):** An access point (AP) consists of a Wi-Fi that is advertising a network name (known as a Service Set Identifier, or SSID). Users who connect to this network will typically find their traffic bridged to a local-area network (LAN) wired network (like Ethernet) for communication to the larger network or even the internet.

SOURCE CODE:

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

// 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");
```

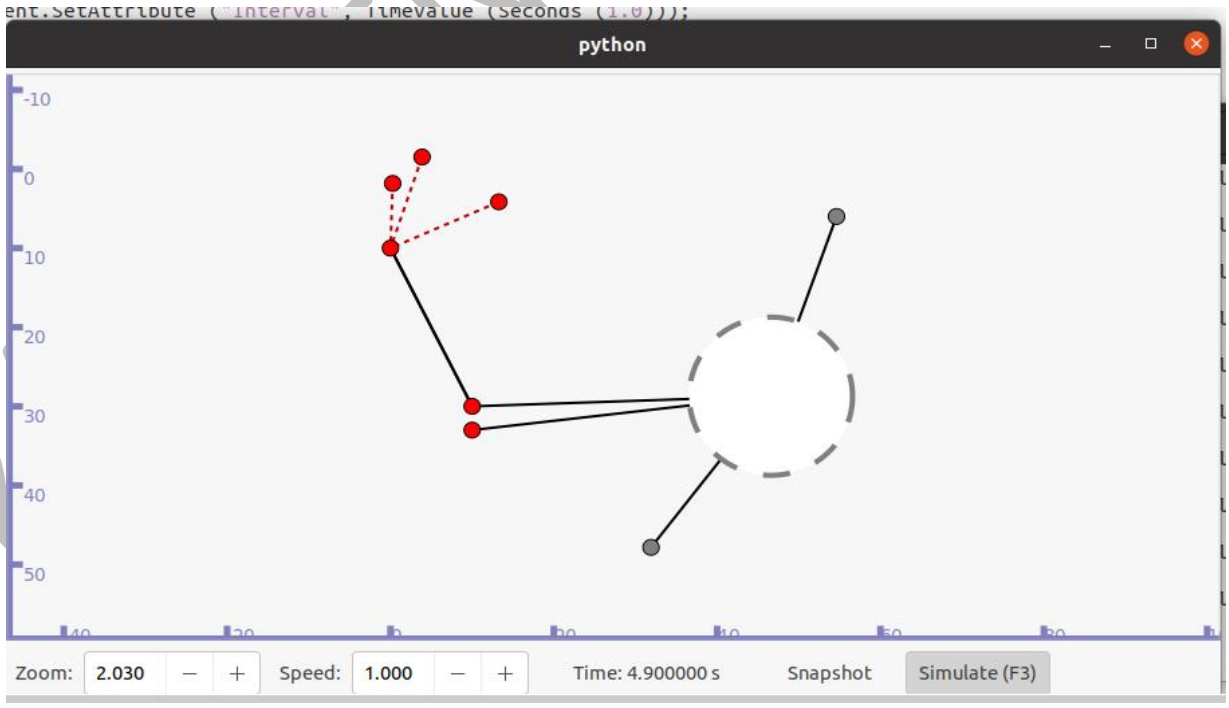
```
int
main (int argc, char *argv[])
{
 bool verbose = true;
 uint32_t nCsma = 3;
 uint32_t nWifi = 3;
 bool tracing = false;

 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),
```

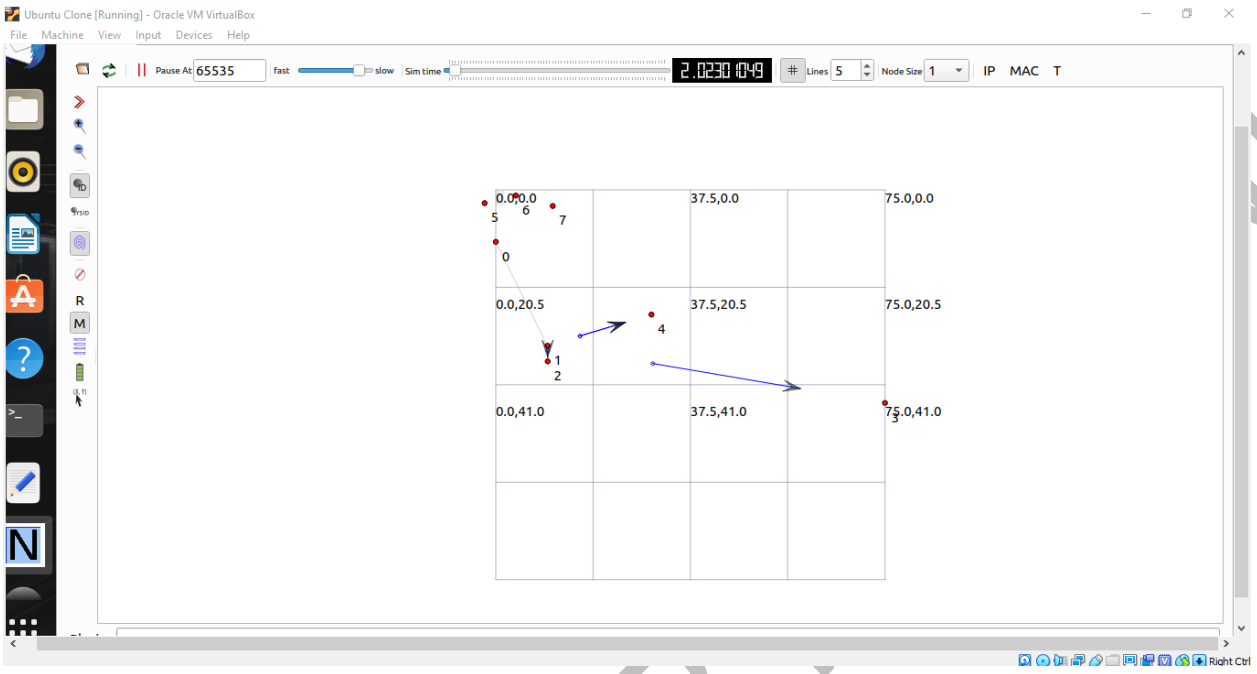
```
                "DeltaY", DoubleValue (10.0),
                "GridWidth", UintegerValue (3),
                "LayoutType", StringValue ("RowFirst"));

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

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

 AnimationInterface::SetConstantPosition (p2pNodes.Get (1), 10, 30);
 AnimationInterface::SetConstantPosition (csmaNodes.Get (1), 10, 33);

 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 ();

 Simulator::Stop (Seconds (10.0));

 if (tracing == true)
  {
    pointToPoint.EnablePcapAll ("third");
    phy.EnablePcap ("third", apDevices.Get (0));
    csma.EnablePcap ("third", csmaDevices.Get (0), true);
  }
 AnimationInterface anim ("narender-wireless-animation.xml");

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

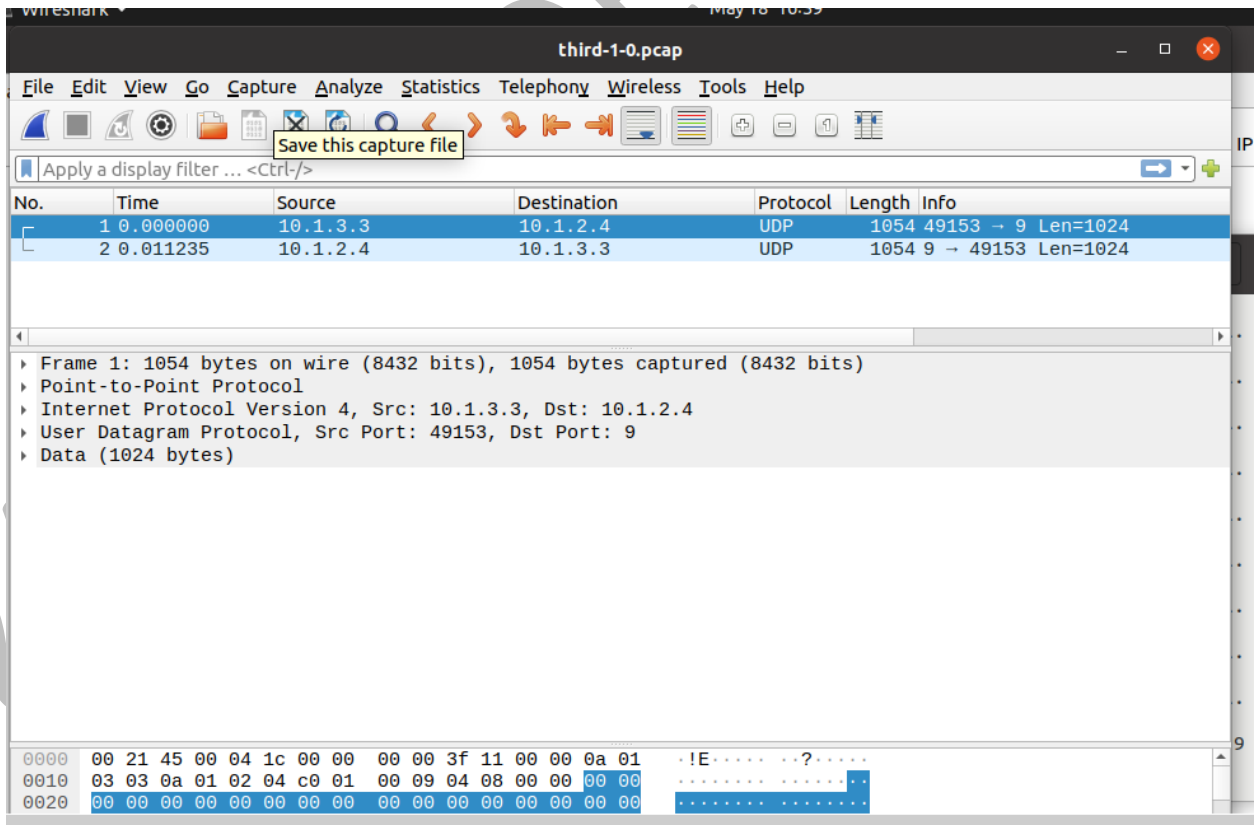
OUTPUT:

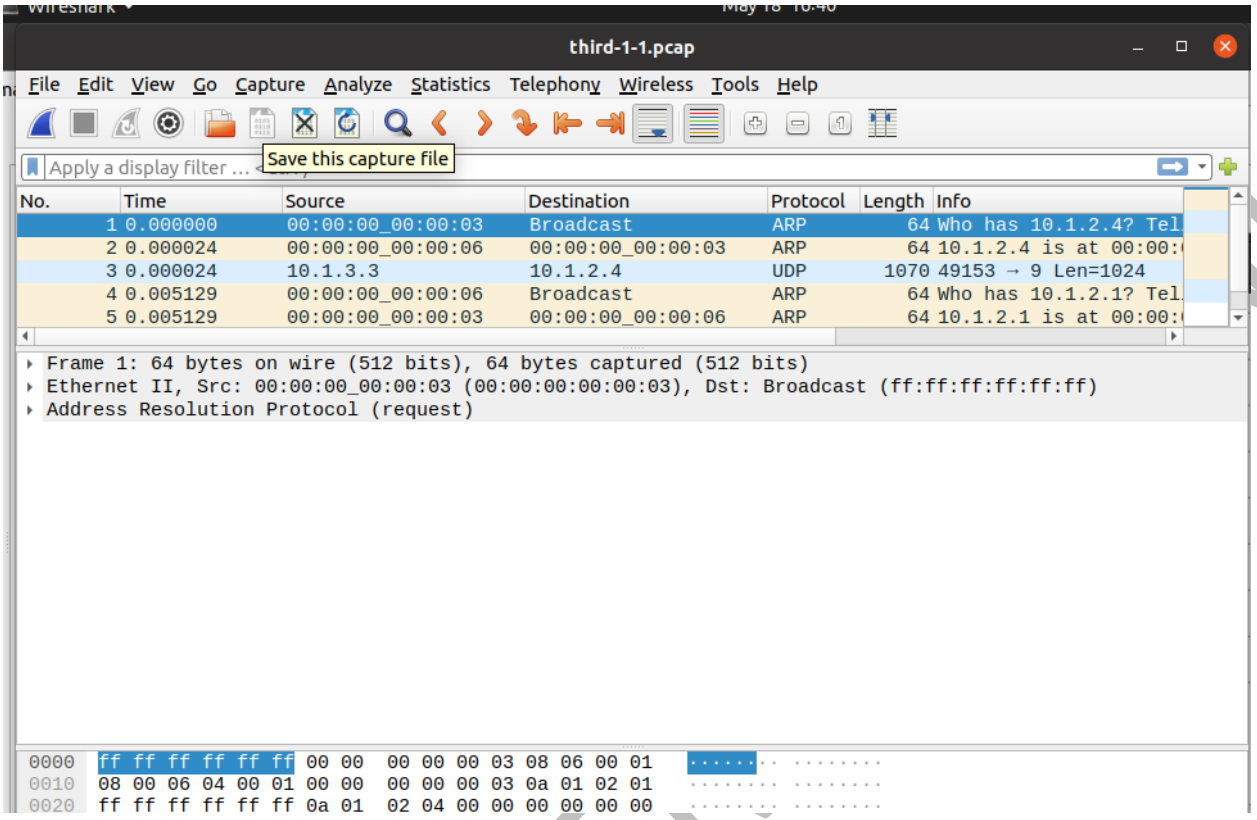


SIMULATION:



WIRESHARK:





CONCLUSION:

From this practical, I have learned how to simulate the wireless networks

AIM: Program to simulate UDP server client.

THEORY:

Understanding UDP server client:

In UDP, the client does not form a connection with the server like in TCP and instead just sends a datagram. Similarly, the server need not accept a connection and just waits for datagrams to arrive. Datagrams upon arrival contain the address of sender which the server uses to send data to the correct client.

In UDP, the client does not form a connection with the server like in TCP and instead, It just sends a datagram. Similarly, the server need not to accept a connection and just waits for datagrams to arrive. We can call a function called **connect()** in UDP but it does not result anything like it does in TCP. There is no 3-way handshake. It just checks for any immediate errors and store the peer's IP address and port number. connect() is storing peers address so no need to pass **server address** and **server address length** arguments in **send to()**. The entire process can be broken down into following steps **UDP Server:**

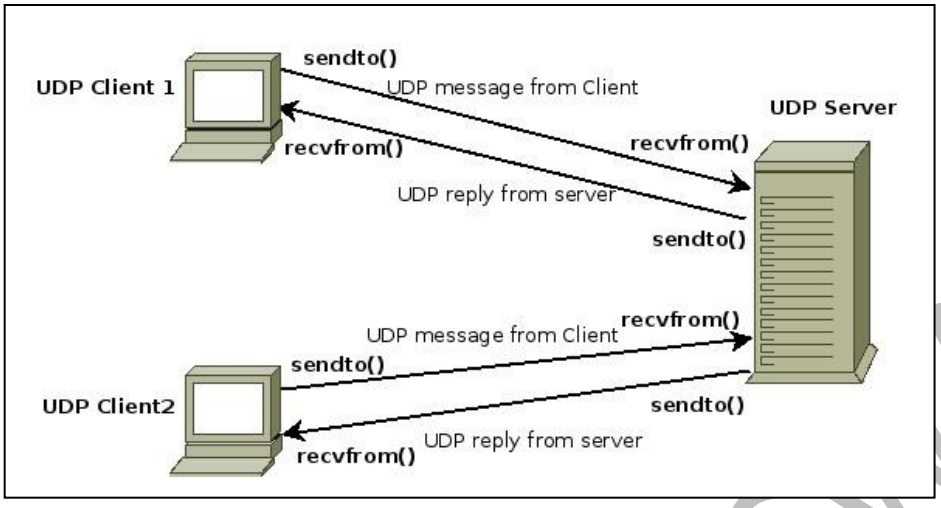
1. Create UDP socket.
2. Bind the socket to server address.
3. Wait until datagram packet arrives from client.
4. Process the datagram packet and send a reply to client.
5. Go back to Step 3.

UDP Client:

1. Create UDP socket.
2. Send message to server.
3. Wait until response from server is received.
4. Process reply and go back to step 2, if necessary.
5. Close socket descriptor and exit.

UDP Overview:

UDP is the abbreviation of User Datagram Protocol. UDP makes use of Internet Protocol of the TCP/IP suit. In communications using UDP, a client program sends a message packet to a destination server wherein the destination server also runs on UDP.



UDP the InterSystems IRIS User Datagram Protocol (UDP) binding. Provides two-way message transfer between a server and a large number of clients. UDP is not connection-based; each data packet transmission is an independent event. Provides fast and lightweight data transmission for local packet broadcasts and remote multicasting. Inherently less reliable than TCP. Does not provide message acknowledgement.

Properties of UDP:

- The UDP does not provide guaranteed delivery of message packets. If for some issue in a network if a packet is lost it could be lost forever.

Since there is no guarantee of assured delivery of messages, UDP is considered an unreliable protocol.

- The underlying mechanisms that implement UDP involve no connection-based communication. There is no streaming of data between a UDP server or and an UDP Client.
- An UDP client can send "n" number of distinct packets to an UDP server and it could also receive "n" number of distinct packets as replies from the UDP server.
- Since UDP is connectionless protocol the overhead involved in UDP is less compared to a connection based protocol like TCP.

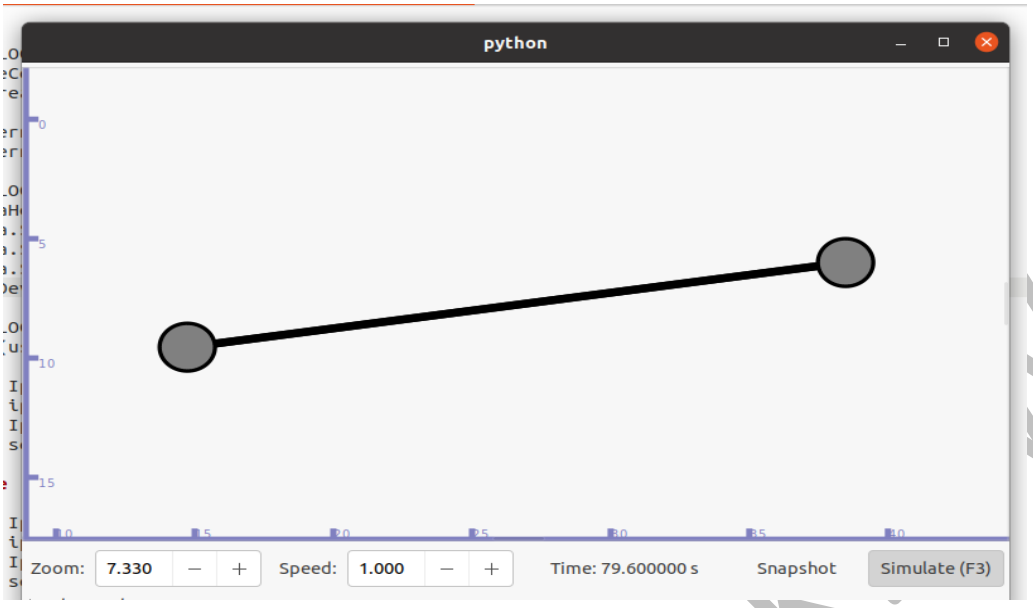
SOURCE CODE:

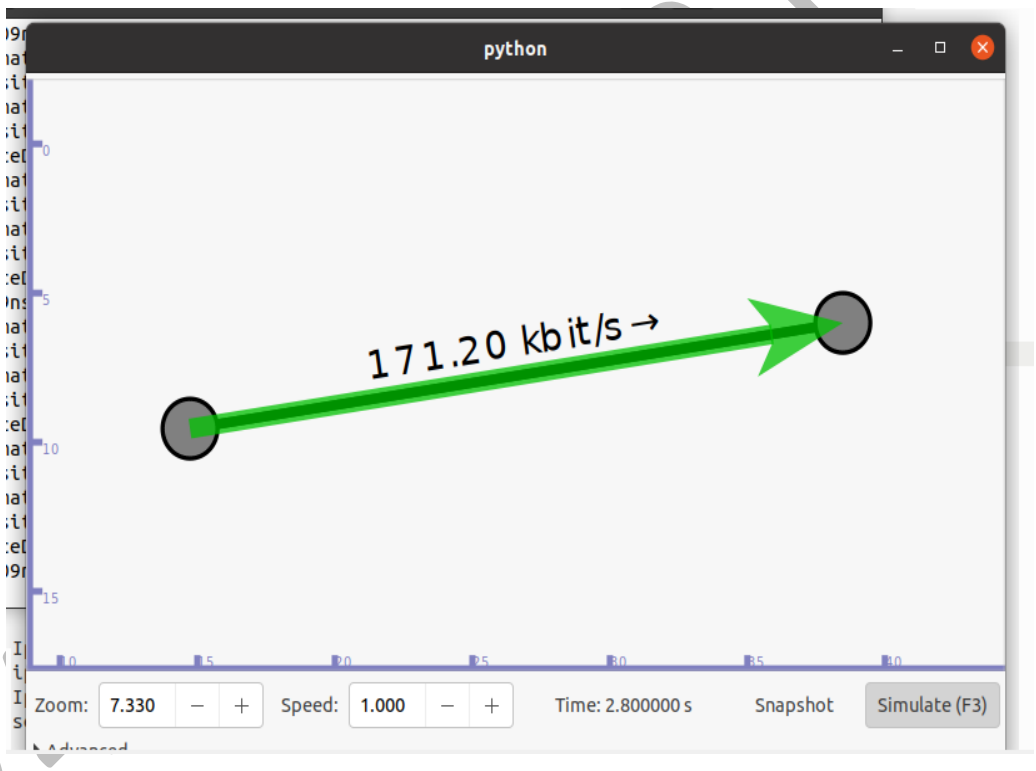
```
// Network topology
//
//       n0   n1
//       |    |
//       =======
//         LAN (CSMA)
//
// - UDP flow from n0 to n1 of 1024 byte packets at intervals of 50 ms
//   - maximum of 320 packets sent (or limited by simulation duration)
//   - option to use IPv4 or IPv6 addressing
```

```
//   - option to disable logging statements

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

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("UdpClientServerExample");

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

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

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

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

 InternetStackHelper internet;
 internet.Install (n);

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

 NS_LOG_INFO ("Assign IP Addresses.");
 if (useV6 == false)
  {
```

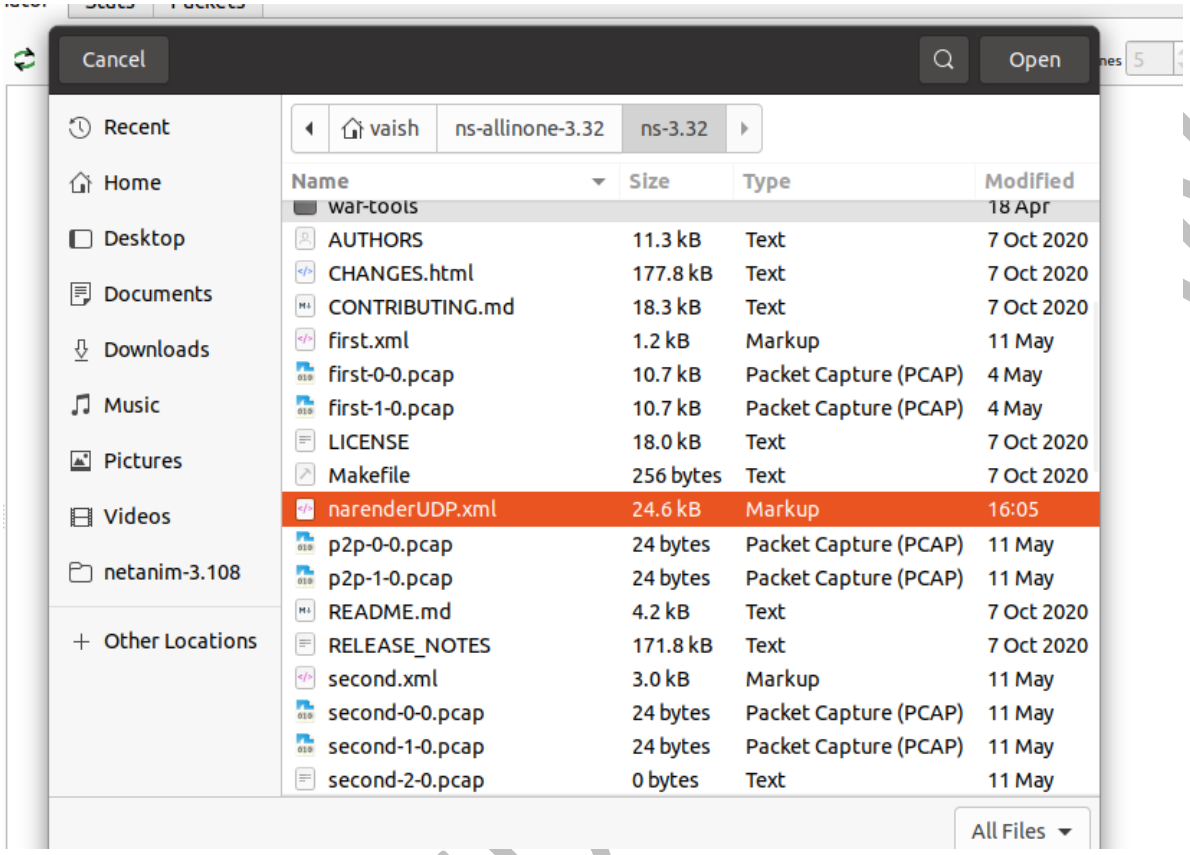
```
    Ipv4AddressHelper ipv4;
    ipv4.SetBase ("10.1.1.0", "255.255.255.0");
    Ipv4InterfaceContainer i = ipv4.Assign (d);
    serverAddress = Address (i.GetAddress (1));
   }
  else
   {
    Ipv6AddressHelper ipv6;
    ipv6.SetBase ("2001:0000:f00d:cafe::", Ipv6Prefix (64));
    Ipv6InterfaceContainer i6 = ipv6.Assign (d);
    serverAddress = Address(i6.GetAddress (1,1));
   }

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

  NS_LOG_INFO ("Create UdpClient application on node 0 to send to node 1.");
  uint32_t MaxPacketSize = 1024;
  Time interPacketInterval = Seconds (0.05);
  uint32_t maxPacketCount = 320;
  UdpClientHelper client (serverAddress, port);
  client.SetAttribute ("MaxPackets", UintegerValue (maxPacketCount));
  client.SetAttribute ("Interval", TimeValue (interPacketInterval));
  client.SetAttribute ("PacketSize", UintegerValue (MaxPacketSize));
  apps = client.Install (n.Get (0));
  apps.Start (Seconds (2.0));
  apps.Stop (Seconds (10.0));

  csma.EnablePcapAll ("udp");

  AnimationInterface anim("narenderUDP.xml");
  NS_LOG_INFO ("Run Simulation.");
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");
}
```

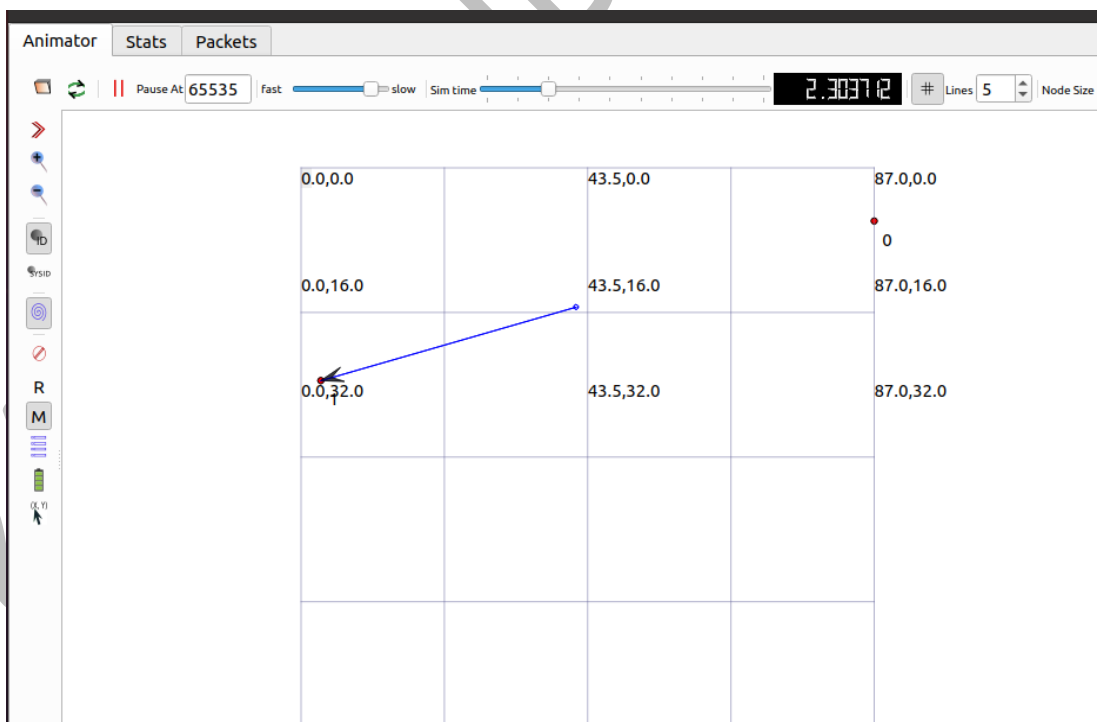
OUTPUT:


python
Zoom: 7.330
Speed: 1.000
Time: 79.600000 s
Snapshot
Simulate (F3)

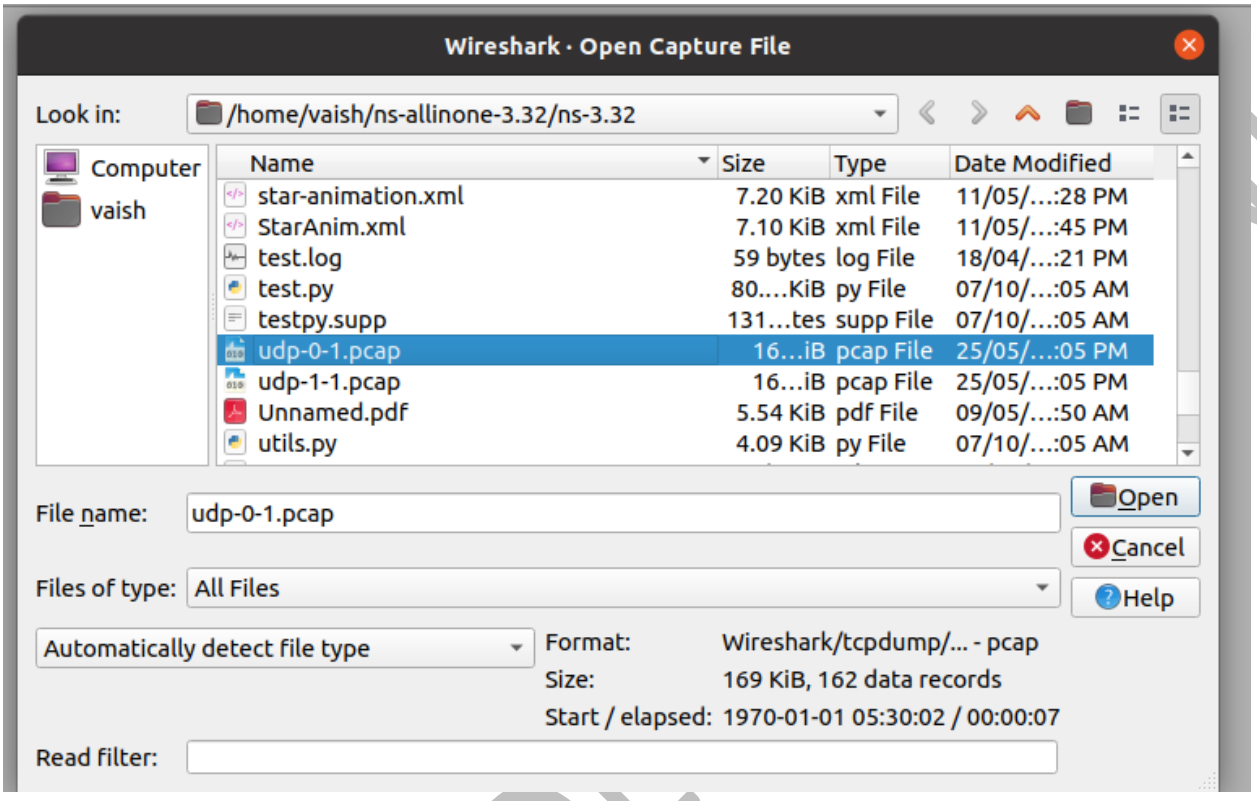

python
171.20 kbit/s →
Zoom: 7.330
Speed: 1.000
Time: 2.800000 s
Snapshot
Simulate (F3)

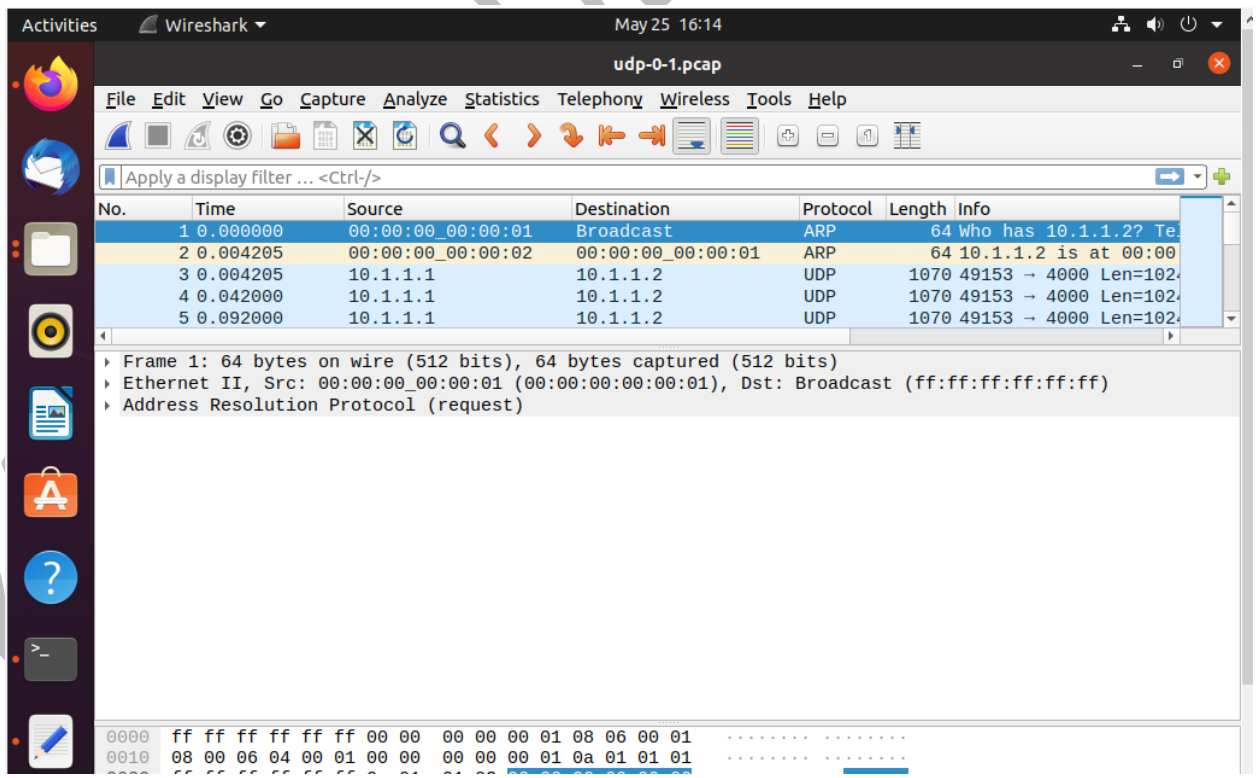
NetAnim:

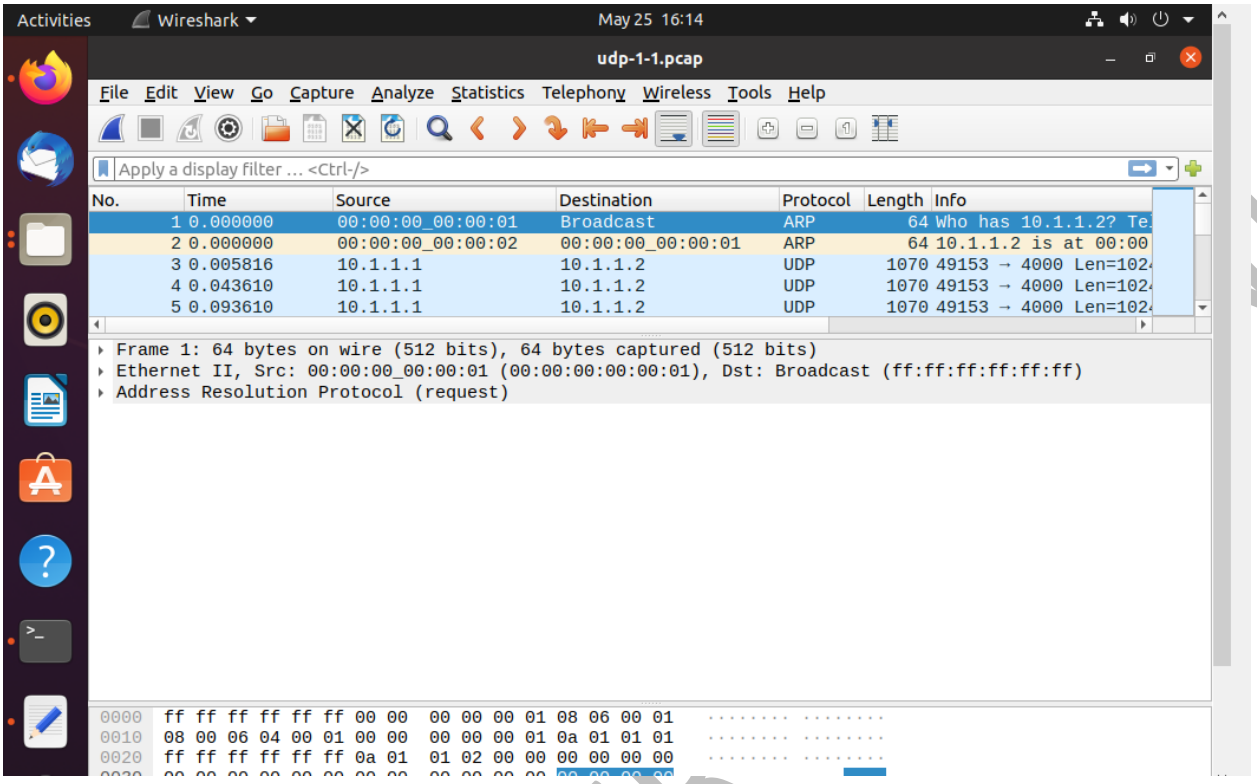

Cancel
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
Other Locations
vaish
ns-allinone-3.32
ns-3.32
Name
Size
Type
Modified
AUTHORS 11.3 kB Text 7 Oct 2020
CHANGES.html 177.8 kB Text 7 Oct 2020
CONTRIBUTING.md 18.3 kB Text 7 Oct 2020
first.xml 1.2 kB Markup 11 May
first-0-0.pcap 10.7 kB Packet Capture (PCAP) 4 May
first-1-0.pcap 10.7 kB Packet Capture (PCAP) 4 May
LICENSE 18.0 kB Text 7 Oct 2020
Makefile 256 bytes Text 7 Oct 2020
narenderUDP.xml 24.6 kB Markup 16:05
p2p-0-0.pcap 24 bytes Packet Capture (PCAP) 11 May
p2p-1-0.pcap 24 bytes Packet Capture (PCAP) 11 May
README.md 4.2 kB Text 7 Oct 2020
RELEASE_NOTES 171.8 kB Text 7 Oct 2020
second.xml 3.0 kB Markup 11 May
second-0-0.pcap 24 bytes Packet Capture (PCAP) 11 May
second-1-0.pcap 24 bytes Packet Capture (PCAP) 11 May
second-2-0.pcap 0 bytes Text 11 May
All Files


Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
2.303712
Lines 5
Node Size
0.0,0.0
43.5,0.0
87.0,0.0
0
0.0,16.0
43.5,16.0
87.0,16.0
0.0,32.0
43.5,32.0
87.0,32.0

WIRESHARK:


Wireshark · Open Capture File
Look in: /home/vaish/ns-allinone-3.32/ns-3.32
Computer
vaish
Name Size Type Date Modified
star-animation.xml 7.20 KiB xml File 11/05/...:28 PM
StarAnim.xml 7.10 KiB xml File 11/05/...:45 PM
test.log 59 bytes log File 18/04/...:21 PM
test.py 80....KiB py File 07/10/...:05 AM
testpy.supp 131...tes supp File 07/10/...:05 AM
udp-0-1.pcap 16...iB pcap File 25/05/...:05 PM
udp-1-1.pcap 16...iB pcap File 25/05/...:05 PM
Unnamed.pdf 5.54 KiB pdf File 09/05/...:50 AM
utils.py 4.09 KiB py File 07/10/...:05 AM
File name: udp-0-1.pcap
Open
Cancel
Files of type: All Files
Help
Automatically detect file type
Format: Wireshark/tcpdump/... - pcap
Size: 169 KiB, 162 data records
Start / elapsed: 1970-01-01 05:30:02 / 00:00:07
Read filter:


Activities
Wireshark
May 25 16:14
udp-0-1.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000 00:00:00_00:00:01 Broadcast ARP 64 Who has 10.1.1.2? Te
2 0.004205 00:00:00_00:00:02 00:00:00_00:00:01 ARP 64 10.1.1.2 is at 00:00
3 0.004205 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=102
4 0.042000 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=102
5 0.092000 10.1.1.1 10.1.1.2 UDP 1070 49153 → 4000 Len=102
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



CONCLUSION:

From this practical, I have successfully implemented the simulation of UDP server client.

AIM: PROGRAM TO SIMULATE DHCP SERVER AND N CLIENTS.

THEORY:

Why use DHCP?

Every device on a TCP/IP-based network must have a unique unicast IP address to access the network and its resources. Without DHCP, IP addresses for new computers or computers that are moved from one subnet to another must be configured manually; IP addresses for computers that are removed from the network must be manually reclaimed.

With DHCP, this entire process is automated and managed centrally. The DHCP server maintains a pool of IP addresses and leases an address to any DHCP-enabled client when it starts up on the network. Because the IP addresses are dynamic (leased) rather than static (permanently assigned), addresses no longer in use are automatically returned to the pool for reallocation.

The network administrator establishes DHCP servers that maintain TCP/IP configuration information and provide address configuration to DHCP-enabled clients in the form of a lease offer. The DHCP server stores the configuration information in a database that includes:

- Valid TCP/IP configuration parameters for all clients on the network.
- Valid IP addresses, maintained in a pool for assignment to clients, as well as excluded addresses.
- Reserved IP addresses associated with particular DHCP clients. This allows consistent assignment of a single IP address to a single DHCP client.
- The lease duration, or the length of time for which the IP address can be used before a lease renewal is required.

A DHCP-enabled client, upon accepting a lease offer, receives:

- A valid IP address for the subnet to which it is connecting.
- Requested DHCP options, which are additional parameters that a DHCP server is configured to assign to clients. Some examples of DHCP options are Router (default gateway), DNS Servers, and DNS Domain Name.

Benefits of DHCP:

DHCP provides the following benefits.

- **Reliable IP address configuration**: DHCP minimizes configuration errors caused by manual IP address configuration, such as typographical errors, or address conflicts caused by the assignment of an IP address to more than one computer at the same time.
- **Reduced network administration**: DHCP includes the following features to reduce network administration:
- Centralized and automated TCP/IP configuration.
- The ability to define TCP/IP configurations from a central location.
- The ability to assign a full range of additional TCP/IP configuration values by means of DHCP options.

- The efficient handling of IP address changes for clients that must be updated frequently, such as those for portable devices that move to different locations on a wireless network.
- The forwarding of initial DHCP messages by using a DHCP relay agent, which eliminates the need for a DHCP server on every subnet.

DHCP Server:

A DHCP Server is a network server that automatically provides and assigns IP addresses, default gateways and other network parameters to client devices. It relies on the standard protocol known as Dynamic Host Configuration Protocol or DHCP to respond to broadcast queries by clients.

A DHCP server automatically sends the required network parameters for clients to properly communicate on the network. Without it, the network administrator has to manually set up every client that joins the network, which can be cumbersome, especially in large networks. DHCP servers usually assign each client with a unique dynamic IP address, which changes when the client's lease for that IP address has expired.

DHCP Client:

The DHCP protocol has two functions with regard to the client. It delivers sufficient information to clients for them to establish an endpoint for network communications, and it supplies other parameters needed by system- and application-level software.

To perform the first function, the DHCP protocol supplies an IP address valid for the network attached to the client's hardware interface. The right to use this IP address is given to the client for a finite period of time, called a lease. This differs from the traditional static configuration. If the client wants to use the IP address for a period of time longer than the original lease, it must periodically negotiate a lease extension with the server through DHCP. When the client no longer needs the IP address, the user of the machine can relinquish its lease, returning it to the pool of available IP addresses. Otherwise, the IP address is reclaimed automatically when its lease expires.

The implementation of the client side of the DHCP protocol on Solaris must meet several criteria. Bootstrapping a Sun workstation is a complex process because of the number and diversity of services that must be configured and invoked. Any DHCP solution must coexist with other methods already in use, particularly the Reverse Address Resolution Protocol (RARP) and static configuration. It must know that the superuser can change the address of a network interface after the workstation has booted. It must be able to configure multiple interfaces. And it must respond to human control and be able to report on the status and provide the statistics of the protocol.

SOURCE CODE:

```
/*
*               Network layout:
 *
```

```
*               R0 is a DHCP server. The DHCP server announced R1 as the default
                router.
*               Nodes N1 will send UDP Echo packets to node A.
 *
 *
*                ┌-----------------------------------------------┐
*                | DHCP Clients                                  |
*                |                                  172.30.0.14  |
*                |                                  DHCP static  |
*                |   ┌──────┐       ┌──────┐       ┌──────┐      |
*                |   |  N0  |       |  N1  |       |  N2  |      |
                 ┌──────┐
*                |   └──────┘       └──────┘       └──────┘      |   ┌──┐
                A   |
*                |      |             |              |     |      |
                 └──────┘
*                └------|-------------|--------------|-----┘      |
                172.30.1.2
*               DHCP Server           |              |            |
                |
*                ┌──────┐       |             |          |     ┌──────┐
                |
*                |  R0  |-------|-------------|----------|-----|  R1
                 |──────┘
*                └──────┘
                 └──────┘172.30.1.1
*               172.30.0.12                              172.30.0.17
 *
*               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/network-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"
```

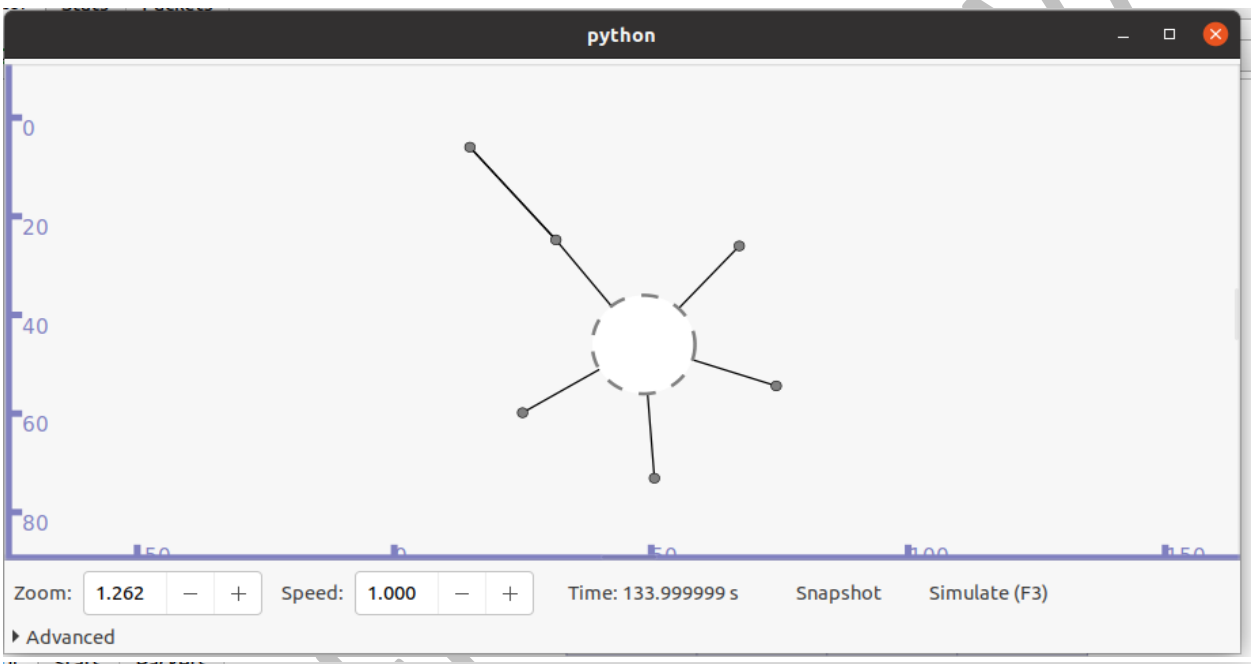
```
#include "ns3/netanim-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("DhcpExample");

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

 bool verbose = true;
 bool tracing = true;
 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));
```

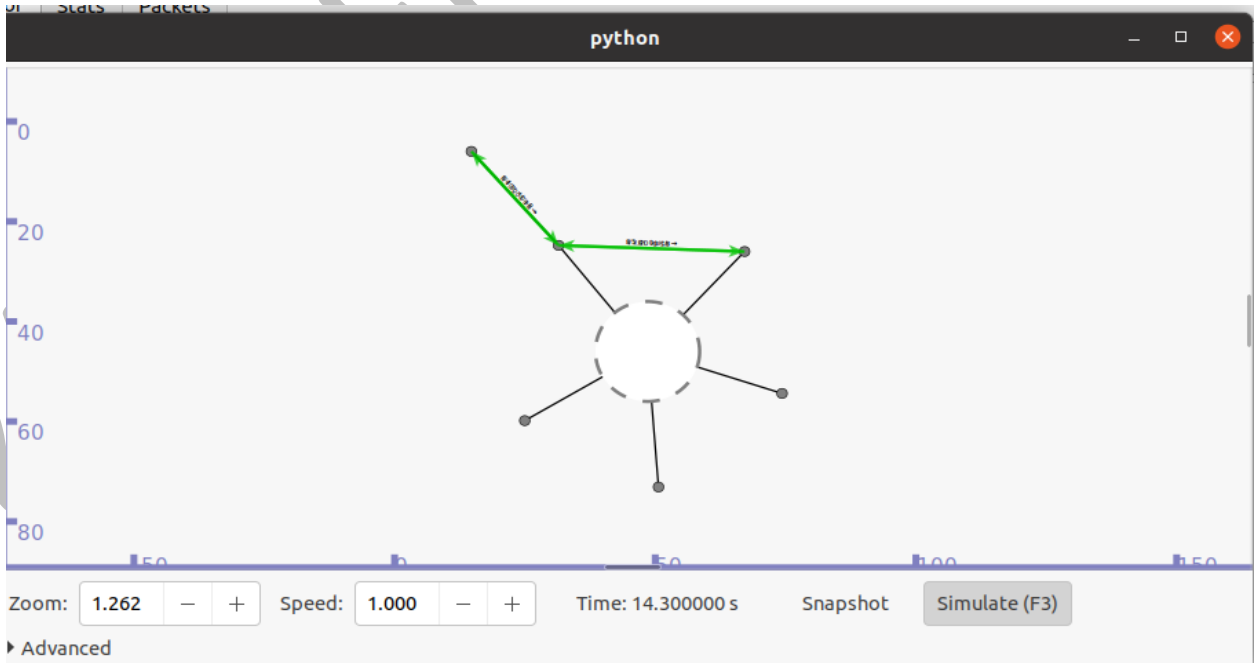
```
NetDeviceContainer devNet = csma.Install (net);

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"),
```

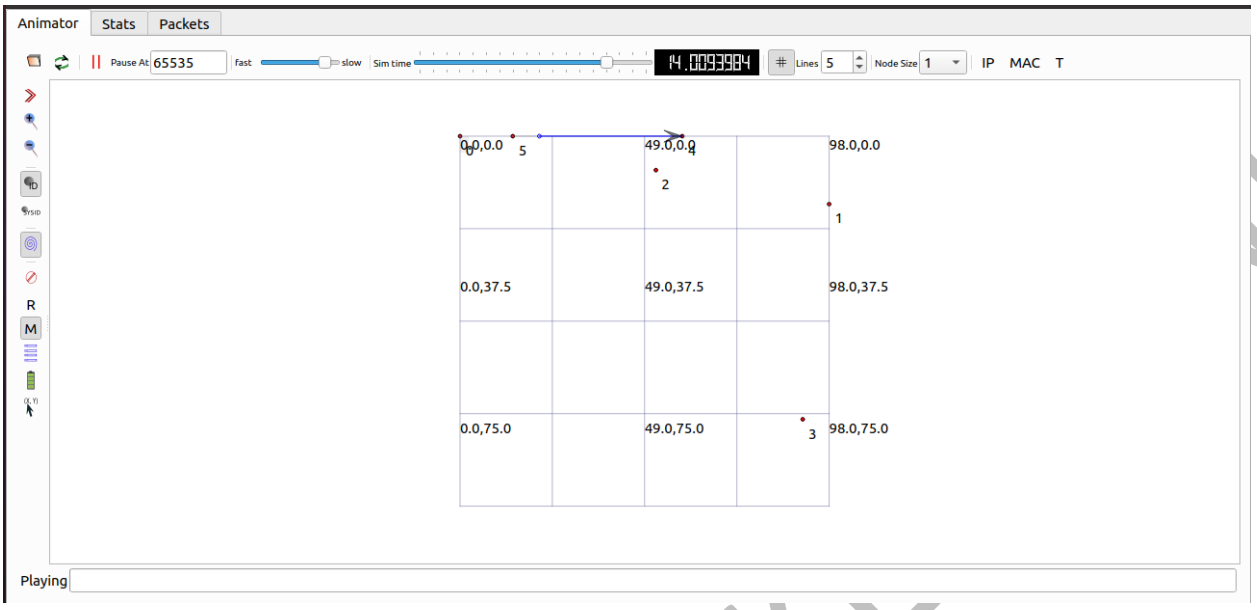
```
                                Ipv4Address ("172.30.0.0"), Ipv4Mask ("/24"),
                                Ipv4Address ("172.30.0.10"), Ipv4Address ("172.30.0.15"),
                                Ipv4Address ("172.30.0.17"));

 // 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));

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

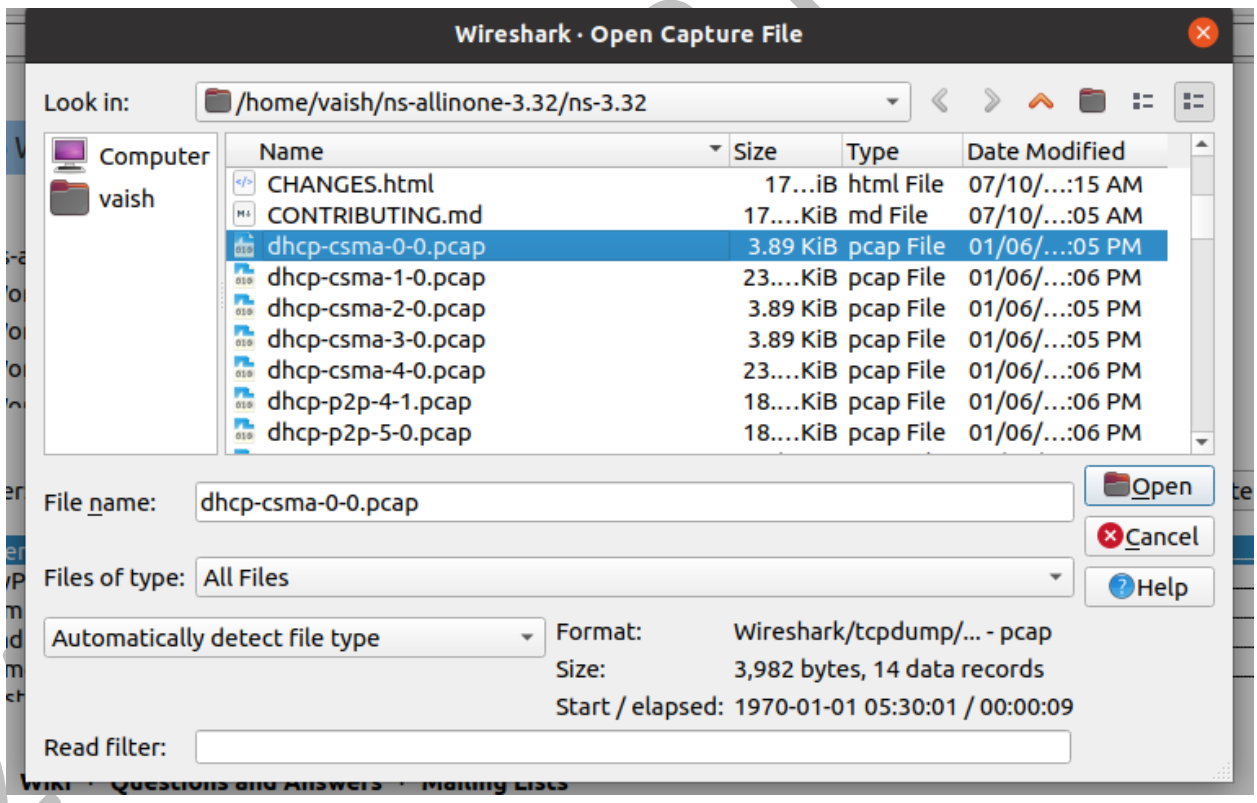
```
  }
 AnimationInterface anim ("narender-dhcp.xml");
 NS_LOG_INFO ("Run Simulation.");
 Simulator::Run ();
 Simulator::Destroy ();
 NS_LOG_INFO ("Done.");
}
```

OUTPUT:






Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
14.0093984
Lines 5
Node Size 1
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
Playing


Wireshark · Open Capture File
Look in:
/home/vaish/ns-allinone-3.32/ns-3.32
Computer
vaish
Name
Size
Type
Date Modified
CHANGES.html 17…iB html File 07/10/…:15 AM
CONTRIBUTING.md 17….KiB md File 07/10/…:05 AM
dhcp-csma-0-0.pcap 3.89 KiB pcap File 01/06/…:05 PM
dhcp-csma-1-0.pcap 23….KiB pcap File 01/06/…:06 PM
dhcp-csma-2-0.pcap 3.89 KiB pcap File 01/06/…:05 PM
dhcp-csma-3-0.pcap 3.89 KiB pcap File 01/06/…:05 PM
dhcp-csma-4-0.pcap 23….KiB pcap File 01/06/…:06 PM
dhcp-p2p-4-1.pcap 18….KiB pcap File 01/06/…:06 PM
dhcp-p2p-5-0.pcap 18….KiB pcap File 01/06/…:06 PM
File name:
dhcp-csma-0-0.pcap
Open
Cancel
Files of type:
All Files
Help
Automatically detect file type
Format: Wireshark/tcpdump/... - pcap
Size: 3,982 bytes, 14 data records
Start / elapsed: 1970-01-01 05:30:01 / 00:00:09
Read filter:

dhcp-csma-0-0.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	290	DHCP Discover - Transaction ID 0xb8e60000
2	0.004986	172.30.0.12	255.255.255.255	DHCP	326	DHCP Offer - Transaction ID 0xb8e60000
3	0.007717	0.0.0.0	255.255.255.255	DHCP	290	DHCP Discover - Transaction ID 0x8ceb0000
4	0.010239	172.30.0.12	255.255.255.255	DHCP	326	DHCP Offer - Transaction ID 0x8ceb0000
5	0.013165	0.0.0.0	255.255.255.255	DHCP	290	DHCP Discover - Transaction ID 0x8d850000

```
Frame 1: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits)
Ethernet II, Src: 00:00:00_00:00:01 (00:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
Dynamic Host Configuration Protocol (Discover)
```

dhcp-csma-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	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	290	DHCP Discover - Transaction ID 0x8ceb0000
2	0.002464	0.0.0.0	255.255.255.255	DHCP	290	DHCP Discover - Transaction ID 0xb8e60000
3	0.004986	172.30.0.12	255.255.255.255	DHCP	326	DHCP Offer - Transaction ID 0xb8e60000
4	0.010239	172.30.0.12	255.255.255.255	DHCP	326	DHCP Offer - Transaction ID 0x8ceb0000
5	0.013165	0.0.0.0	255.255.255.255	DHCP	290	DHCP Discover - Transaction ID 0x8d850000

```
Frame 1: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits)
Ethernet II, Src: 00:00:00_00:00:02 (00:00:00:00:00:02), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
Dynamic Host Configuration Protocol (Discover)
```

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	290	DHCP Discover - Transaction ID 0x8d850000
2	0.002464	0.0.0.0	255.255.255.255	DHCP	290	DHCP Discover - Transaction ID 0xb8e60000
3	0.004986	172.30.0.12	255.255.255.255	DHCP	326	DHCP Offer - Transaction ID 0xb8e60000
4	0.007717	0.0.0.0	255.255.255.255	DHCP	290	DHCP Discover - Transaction ID 0x8ceb0000
5	0.010239	172.30.0.12	255.255.255.255	DHCP	326	DHCP Offer - Transaction ID 0x8ceb0000

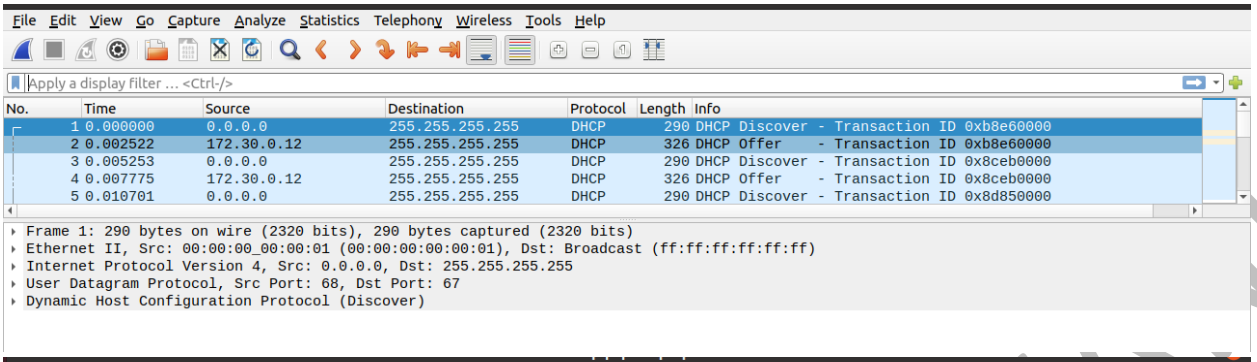
```
Frame 1: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits)
Ethernet II, Src: 00:00:00_00:00:03 (00:00:00:00:00:03), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
Dynamic Host Configuration Protocol (Discover)
```

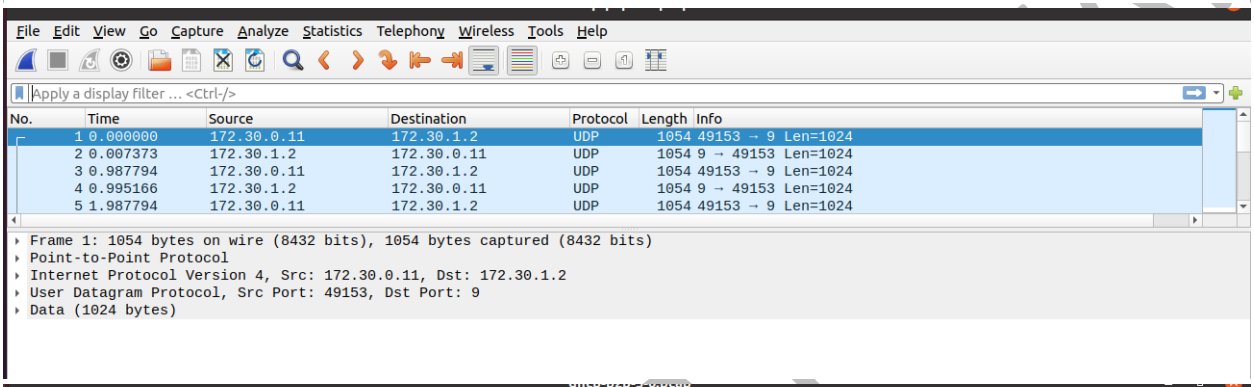
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

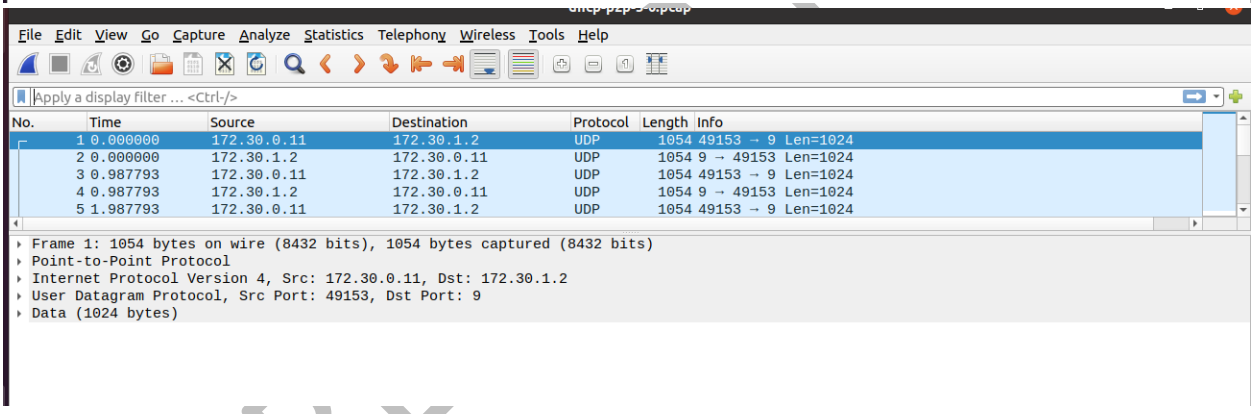
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	290	DHCP Discover - Transaction ID 0xb8e60000
2	0.000000	172.30.0.12	255.255.255.255	DHCP	326	DHCP Offer - Transaction ID 0xb8e60000
3	0.005253	0.0.0.0	255.255.255.255	DHCP	290	DHCP Discover - Transaction ID 0x8ceb0000
4	0.005253	172.30.0.12	255.255.255.255	DHCP	326	DHCP Offer - Transaction ID 0x8ceb0000
5	0.010701	0.0.0.0	255.255.255.255	DHCP	290	DHCP Discover - Transaction ID 0x8d850000

```
Frame 1: 290 bytes on wire (2320 bits), 290 bytes captured (2320 bits)
Ethernet II, Src: 00:00:00_00:00:01 (00:00:00:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 68, Dst Port: 67
Dynamic Host Configuration Protocol (Discover)
```







CONCLUSION:

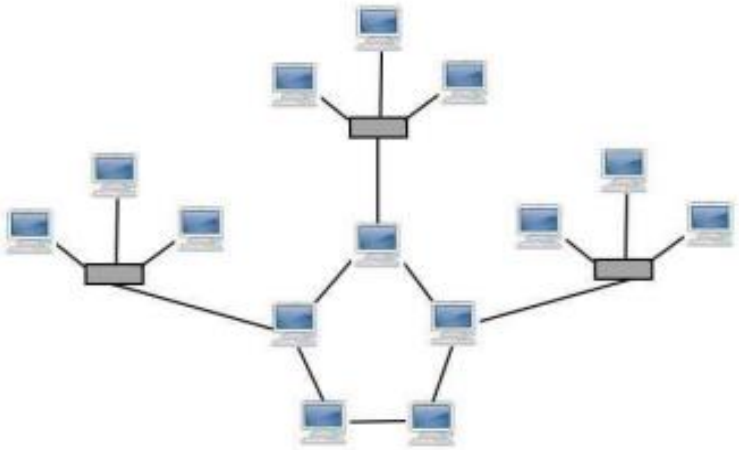
From this practical, I have understood the implementation of dhcp server in ns3 simulator.

AIM: ANIMATE A SIMPLE NETWORK USING NETANIM IN NETWORK SIMULATOR.

THEORY:

Understanding Hybrid Topology (Wireless Network Topology):

A hybrid topology is a kind of network topology that is a combination of two or more network topologies, such as mesh topology, bus topology, and ring topology. Its usage and choice are dependent on its deployments and requirements like the performance of the desired network, and the number of computers, their location. The below figure is describing the structure of hybrid topology that contains more than one topology.



However, a variety of technologies are needed for its physical implementation, and it offers a complex structure. Also, it includes an advantage as increasing flexibility; it can increase fault tolerance, and allows new basic topologies to be added or removed easily. The hybrid topology is more useful when you need to fulfil diversity in Computer Network. In this topology, all network sections can include the configuration of different Network Topology. For instance, you can have a Hybrid network made by two different networks Star Backbone and the Ring Network. You can also use the Star Mesh Hybrid Topology in which if the main backbone gets fail, the entire network will destroy.

SOURCE CODE:

1st: search for star.cc which is located in cd workspace/ns-allinone-3.33/ns-3.33/scratch/bus

2nd: If its not there then create one using cp examples/tutorial/third.cc scratch/Hybrid.cc and paste the following code:

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

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

// 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 ();

AnimationInterface anim ("Hybrid.xml");
anim.SetConstantPosition (p2pNodes.Get(0),10,40);
```

```
 anim.SetConstantPosition (p2pNodes.Get(1),30,40);
 anim.SetConstantPosition (csmaNodes.Get(1),60,40);
 anim.SetConstantPosition (csmaNodes.Get(2),90,40);
 anim.SetConstantPosition (csmaNodes.Get(3),120,40);

 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;
}
```

3rd : cd workspace/ns-allinone-3.33/ns-3.33

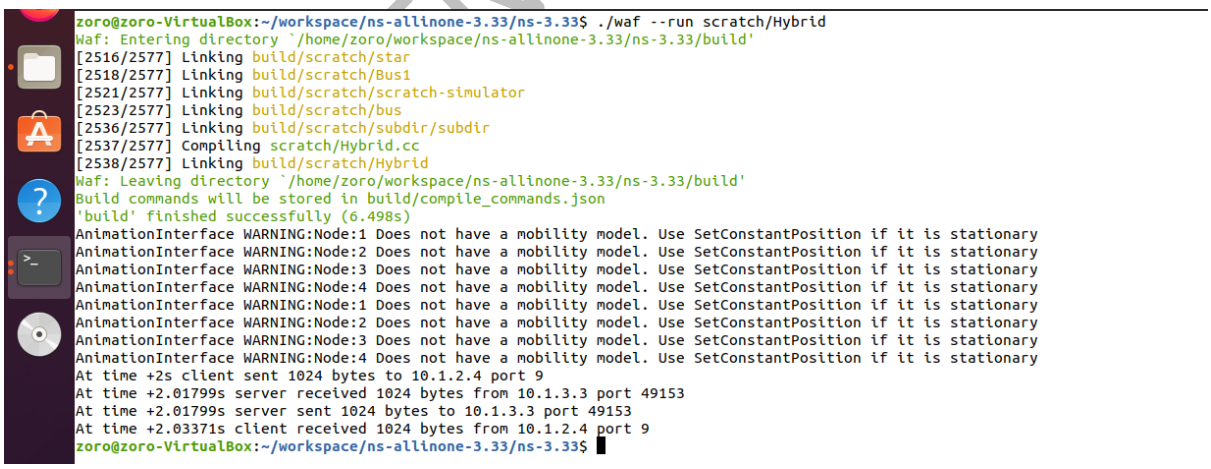
./waf --run scratch/Hybrid

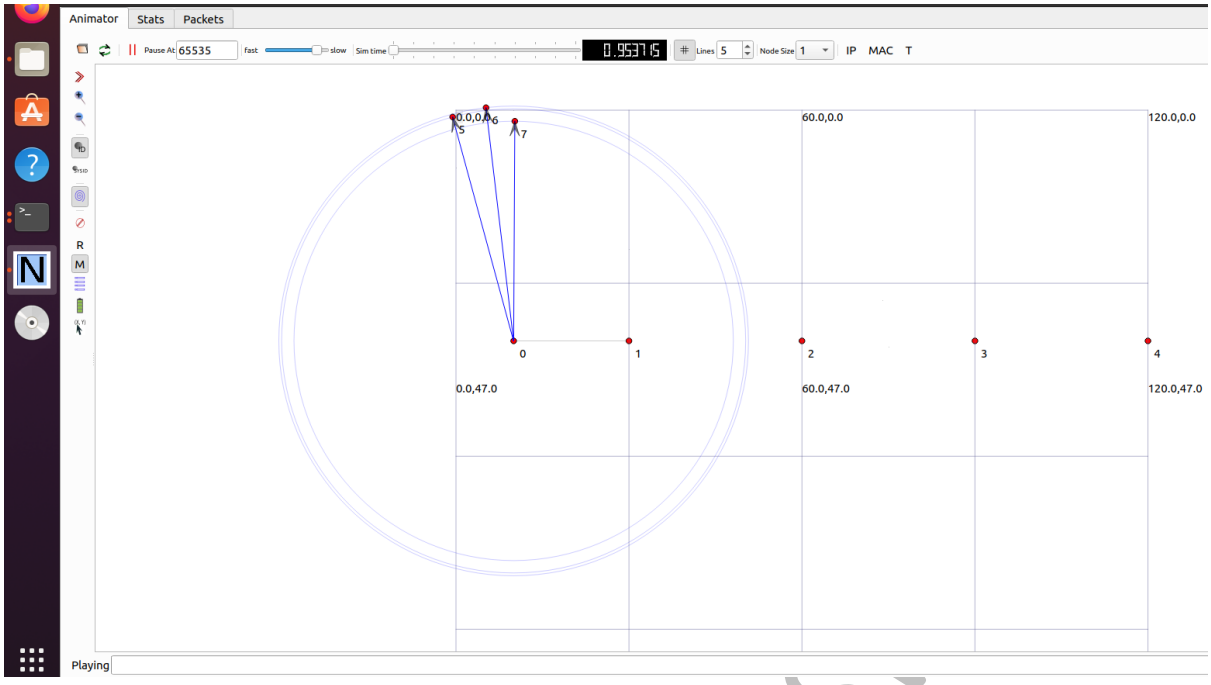
4th : cd

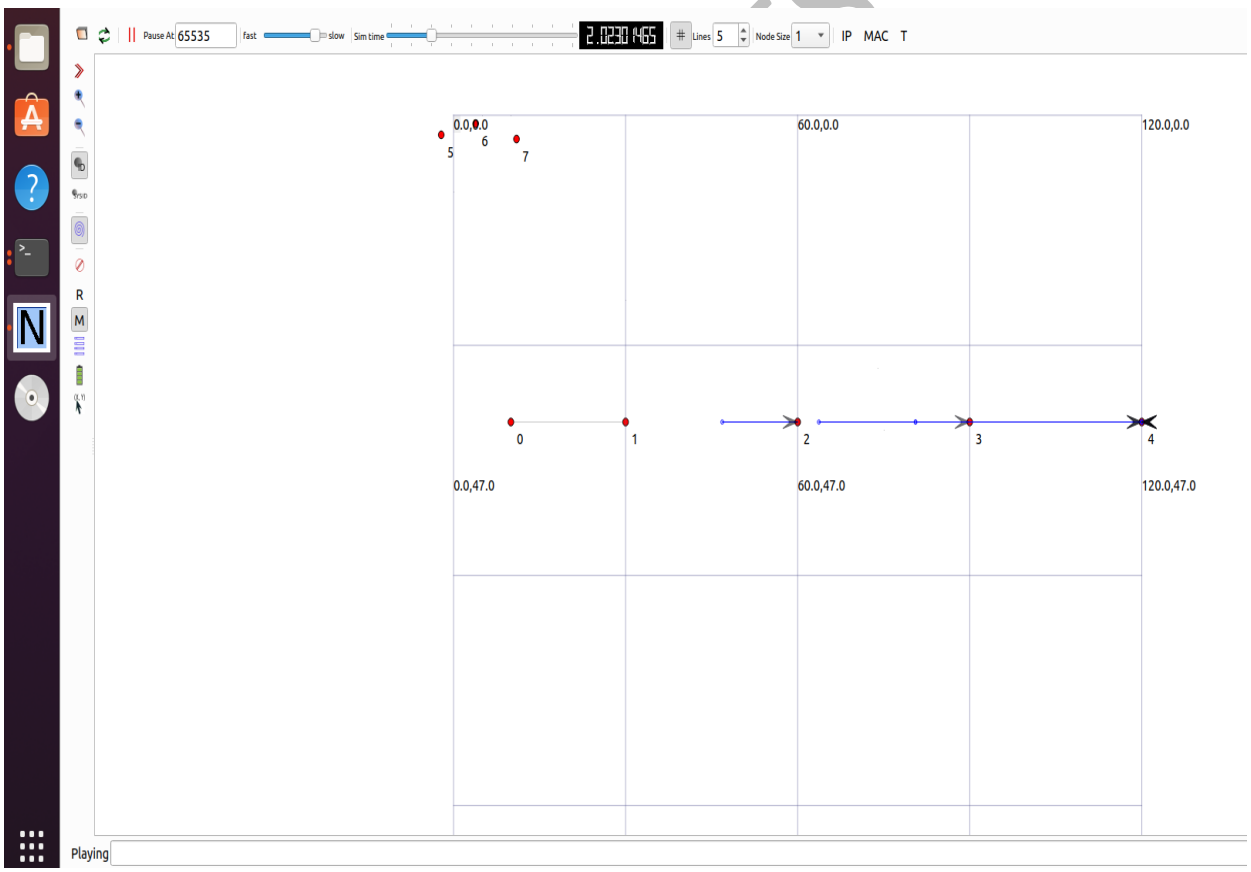
cd workspace/ns-allinone-3.33/netanim-3.108

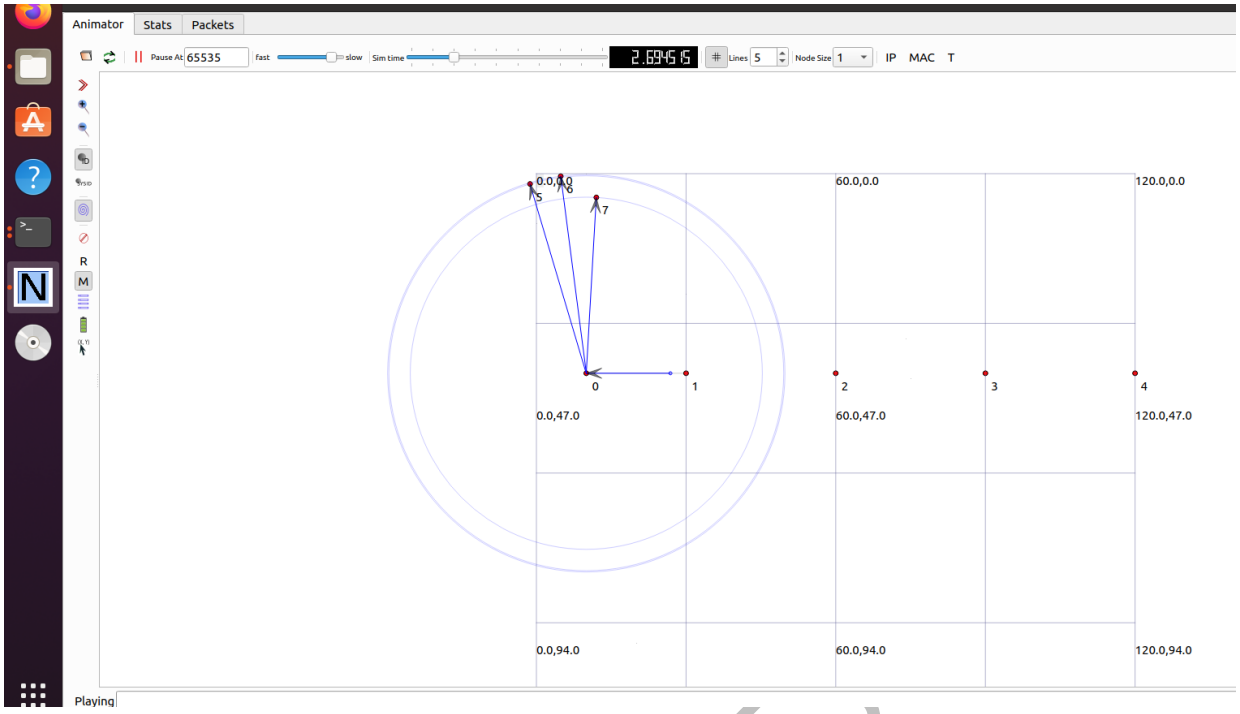
./NetAnim

OUTPUT:




Animator
Stats
Packets
Pause At 65535
fast
slow
Sim time
Lines 5
Node Size 1
IP
MAC
T
0.0,0.0
60.0,0.0
120.0,0.0
0.0,47.0
60.0,47.0
120.0,47.0
0
1
2
3
4
5
6
7
Playing


Pause At 65535
fast
slow
Sim time
Lines 5
Node Size 1
IP
MAC
T
0.0,0.0
60.0,0.0
120.0,0.0
0.0,47.0
60.0,47.0
120.0,47.0
0
1
2
3
4
5
6
7
Playing



CONCLUSION:

From this practical, I have learned how to implement a simple network (hybrid) using NetAnim in Network Simulator executed successfully.

AIM: PROGRAM TO SIMULATE FTP USING TCP PROTOCOL

THEORY:

File Transfer Protocol (FTP):

File transfer protocol (FTP) is a set of rules that computers follow for the transferring of files from one system to another over the internet. It may be used by a business to transfer files from one computer system to another, or websites may use FTP to upload or download files from a website's server.

- File transfer protocol (FTP) is a way to download, upload, and transfer files from one location to another on the internet and between computer systems.
- File transfer protocol (FTP) enables computers on the internet to transfer files back and forth, and is an essential tool for those building and maintaining websites today.
- Many file transfer protocol (FTP) clients are available for free to download, although most websites (and web browsers) that offer downloads already have the FTP built-in, so downloading a separate piece of software isn't always required.

Understanding File Transfer Protocol (FTP):

File transfer protocol is one of many different protocols that dictate how computers behave on the internet. Other such protocols include the Hypertext Transfer Protocol (HTTP), the Internet Message Access Protocol (IMAP), and the Network Time Protocol (NTP). FTP enables computers on the internet to transfer files back and forth, and is an essential tool for those building and maintaining websites today.

In order to use FTP, a user must first download an FTP client (or access an FTP client through a web browser). A client is the software that will allow you to transfer files.

Most web browsers come with FTP clients—possibly via a downloadable extension—that enable users to transfer files from their computer to a server and vice versa. Some users may want to use a third-party FTP client because many of them offer extra features to improve your experience. Examples of FTP clients that are free to download include FileZilla Client, FTP Voyager, WinSCP, CoffeeCup Free FTP, and Core FTP.

Many people have used FTP before without even noticing it. If you have ever downloaded a file from a web page, chances are that you used FTP in the process. The first step for accessing an FTP server to download a file is to log in, which may occur automatically or by manually inputting a username and password. FTP will also require you to access an FTP server through a specific port number.

Once you have accessed the FTP server through your FTP client, you can now transfer files. Not all public FTP servers require you to sign in because some servers enable you to access them anonymously.

Depending on the FTP client you use, there will be different features available that allow you to modify the manner in which you upload and download files. For instance, if you use the free FTP client FileZilla, the program will enable you to set bandwidth limits for files, enabling you to control the speed at which you download or upload files. This can be helpful if you are managing multiple file transfers at once. Other features you may want to look for in an FTP client include public key authentication, the ability to set file compression levels, or tools that enable you to search a server using file masks.

TCP/IP Protocol:

TCP/IP stands for Transmission Control Protocol/Internet Protocol and is a suite of communication protocols used to interconnect network devices on the internet. TCP/IP is also used as a communications protocol in a private computer network (an intranet or extranet). The entire IP suite -- a set of rules and procedures -- is commonly referred to as TCP/IP. TCP and IP are the two main protocols, though others are included in the suite. The TCP/IP protocol suite functions as an abstraction layer between internet applications and the routing and switching fabric.

TCP/IP specifies how data is exchanged over the internet by providing end-to-end communications that identify how it should be broken into packets, addressed, transmitted, routed and received at the destination. TCP/IP requires little central management and is designed to make networks reliable with the ability to recover automatically from the failure of any device on the network.

The two main protocols in the IP suite serve specific functions. TCP defines how applications can create channels of communication across a network. It also manages how a message is assembled into smaller packets before they are then transmitted over the internet and reassembled in the right order at the destination address.

IP defines how to address and route each packet to make sure it reaches the right destination. Each gateway computer on the network checks this IP address to determine where to forward the message.

A subnet mask tells a computer, or other network device, what portion of the IP address is used to represent the network and what part is used to represent hosts, or other computers, on the network.

Network address translation (NAT) is the virtualization of IP addresses. NAT helps improve security and decrease the number of IP addresses an organization needs.

How does TCP/IP work?

TCP/IP uses the client-server model of communication in which a user or machine (a client) is provided a service, like sending a webpage, by another computer (a server) in the network.

Collectively, the TCP/IP suite of protocols is classified as stateless, which means each client request is considered new because it is unrelated to previous requests. Being stateless frees up network paths so they can be used continuously.

The transport layer itself, however, is stateful. It transmits a single message, and its connection remains in place until all the packets in a message have been received and reassembled at the destination.

The TCP/IP model differs slightly from the seven-layer Open Systems Interconnection (OSI) networking model designed after it. The OSI reference model defines how applications can communicate over a network.

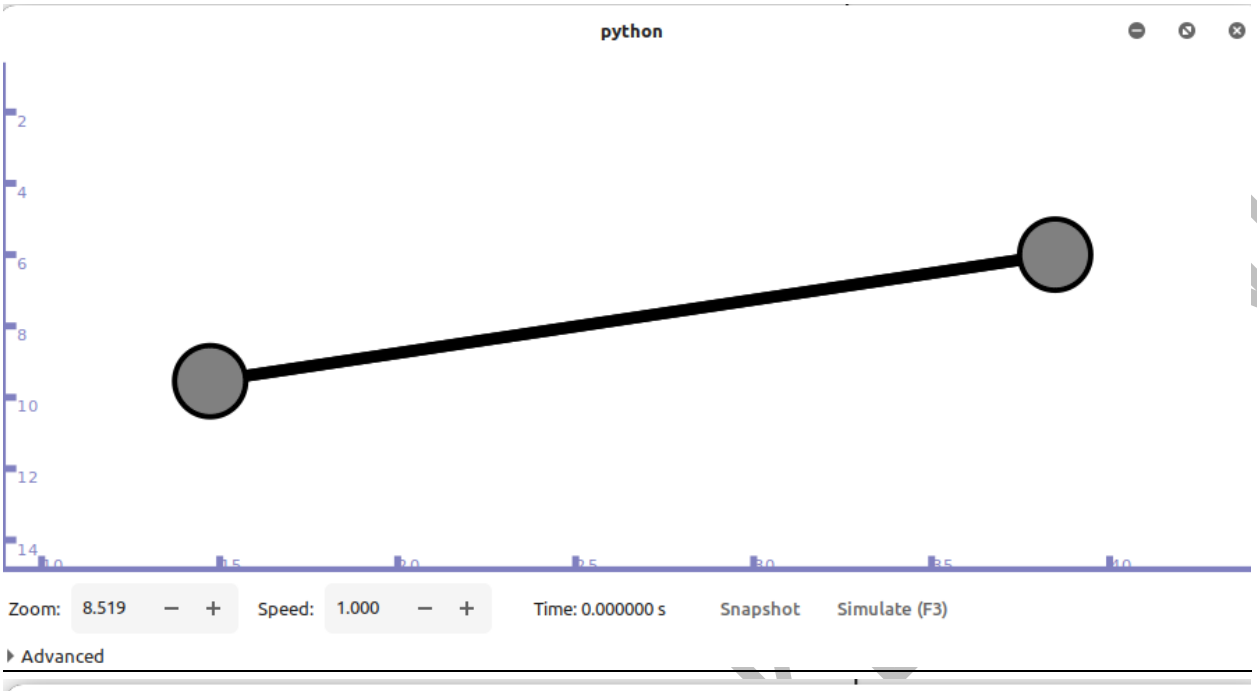
SOURCE CODE:

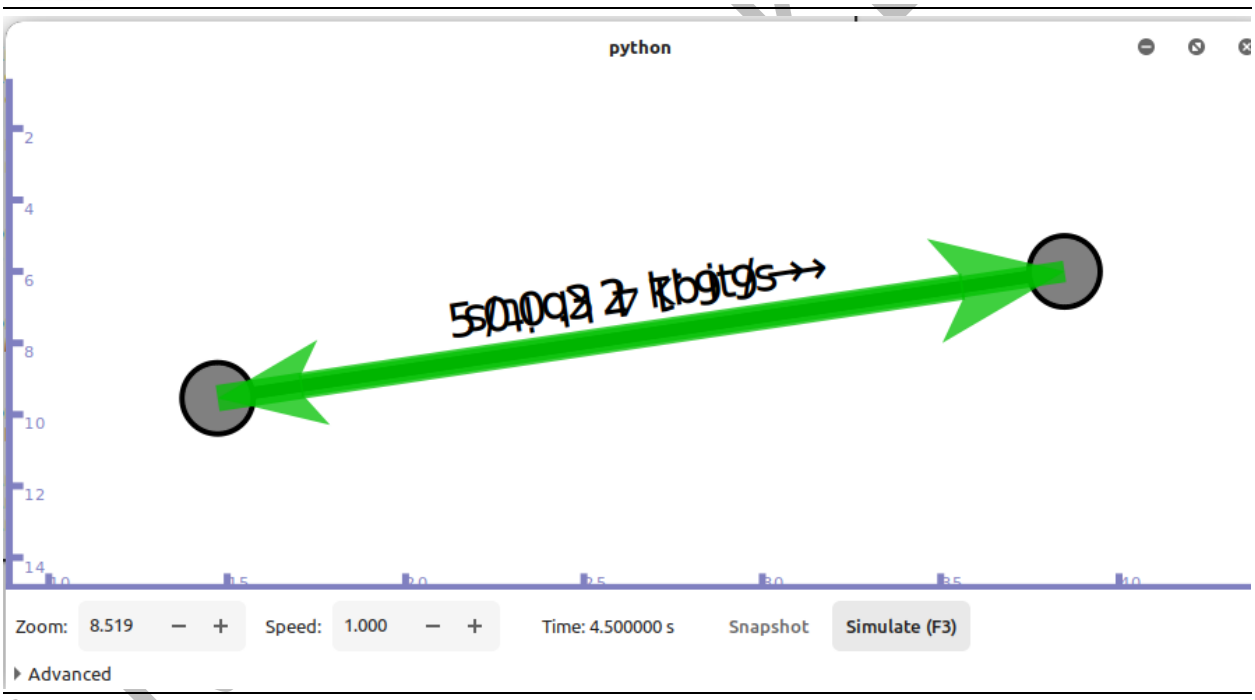
```
// Network topology
//
//       n0 ---------- n1
//            500 Kbps
//             5 ms
//
// - Flow from n0 to n1 using BulkSendApplication.
// - Tracing of queues and packet receptions to file "tcp-bulk-send.tr"
//   and pcap tracing available when tracing is turned on.

#include <string>
#include <fstream>
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/internet-module.h"
#include "ns3/applications-module.h"
#include "ns3/network-module.h"
#include "ns3/packet-sink.h"
#include "ns3/netanim-module.h"
#include "ns3/mobility-module.h"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("TcpBulkSendExample");

Int main (int argc, char *argv[])
{
  bool tracing = true;
  uint32_t maxBytes = 0;

// Allow the user to override any of the defaults at
// run-time, via command-line arguments

  CommandLine cmd (__FILE__);
  cmd.AddValue ("tracing", "Flag to enable/disable tracing", tracing);
  cmd.AddValue ("maxBytes",
                "Total number of bytes for application to send", maxBytes);
  cmd.Parse (argc, argv);

// Explicitly create the nodes required by the topology (shown above).

  NS_LOG_INFO ("Create nodes.");
```

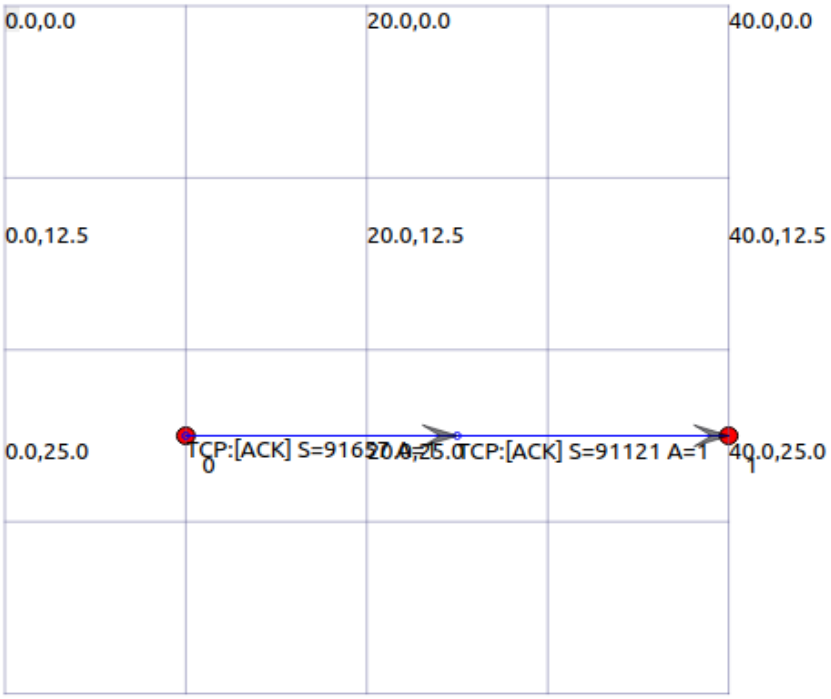
```
  NodeContainer nodes;
  nodes.Create (2);

  NS_LOG_INFO ("Create channels.");

// Explicitly create the point-to-point link required by the topology (shown above).

  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("500Kbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("5ms"));

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

// Install the internet stack on the nodes

  InternetStackHelper internet;
  internet.Install (nodes);

// We've got the "hardware" in place.  Now we need to add IP addresses.

  NS_LOG_INFO ("Assign IP Addresses.");
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer i = ipv4.Assign (devices);

  NS_LOG_INFO ("Create Applications.");

// Create a BulkSendApplication and install it on node 0

  uint16_t port = 9;  // well-known echo port number

  BulkSendHelper source ("ns3::TcpSocketFactory",
                         InetSocketAddress (i.GetAddress (1), port));
  // Set the amount of data to send in bytes.  Zero is unlimited.
  source.SetAttribute ("MaxBytes", UintegerValue (maxBytes));
  ApplicationContainer sourceApps = source.Install (nodes.Get (0));
  sourceApps.Start (Seconds (0.0));
  sourceApps.Stop (Seconds (10.0));

// Create a PacketSinkApplication and install4 it on node 1
  PacketSinkHelper sink ("ns3::TcpSocketFactory",
                         InetSocketAddress (Ipv4Address::GetAny (), port));
  ApplicationContainer sinkApps = sink.Install (nodes.Get (1));
```

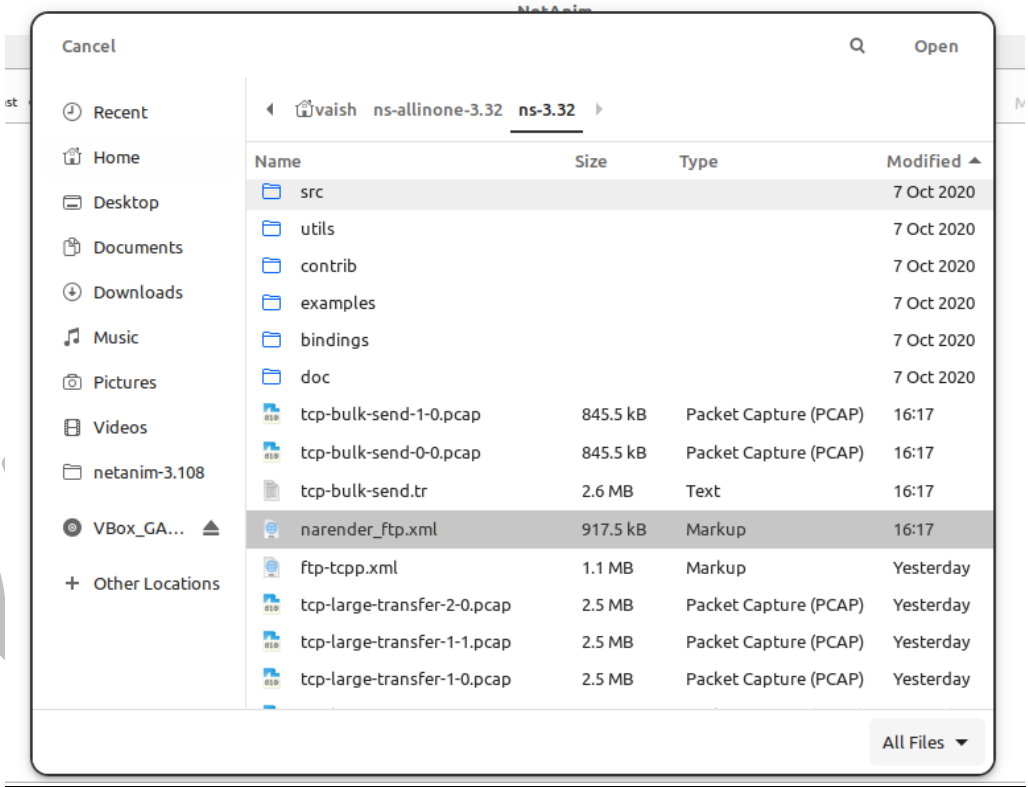
```
  sinkApps.Start (Seconds (0.0));
  sinkApps.Stop (Seconds (10.0));

// Set up tracing if enabled

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

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

  if (tracing)
    {
      AsciiTraceHelper ascii;
      pointToPoint.EnableAsciiAll (ascii.CreateFileStream ("tcp-bulk-send.tr"));
      pointToPoint.EnablePcapAll ("tcp-bulk-send", false);
    }

// Now, do the actual simulation.

  NS_LOG_INFO ("Run Simulation.");
  Simulator::Stop (Seconds (10.0));
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");

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

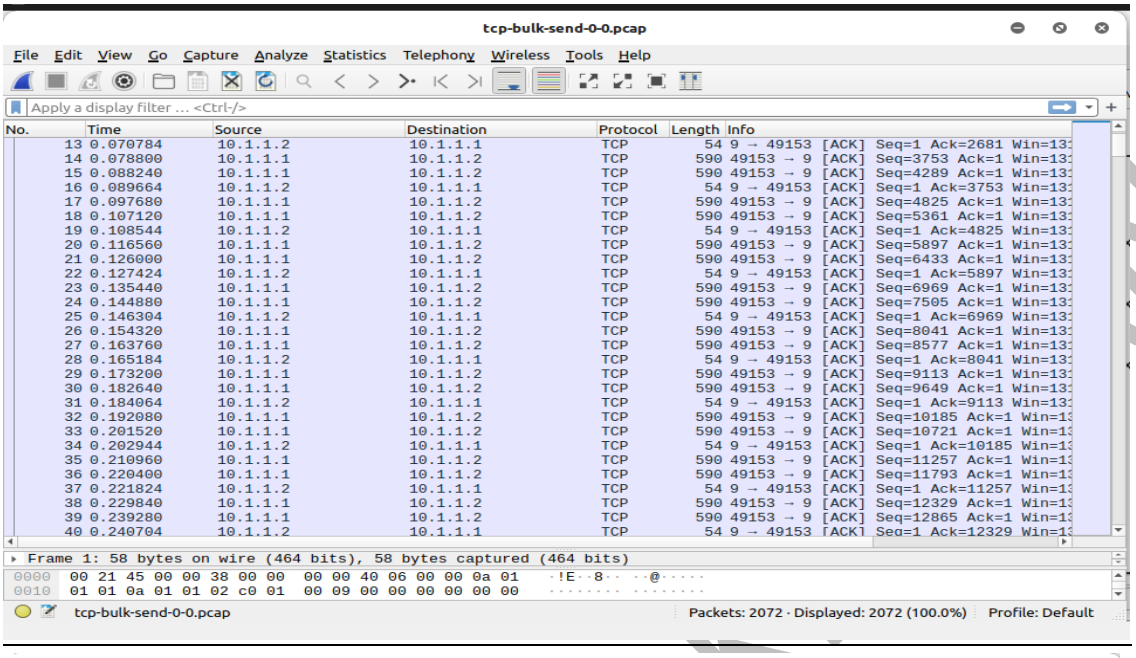
OUTPUT:

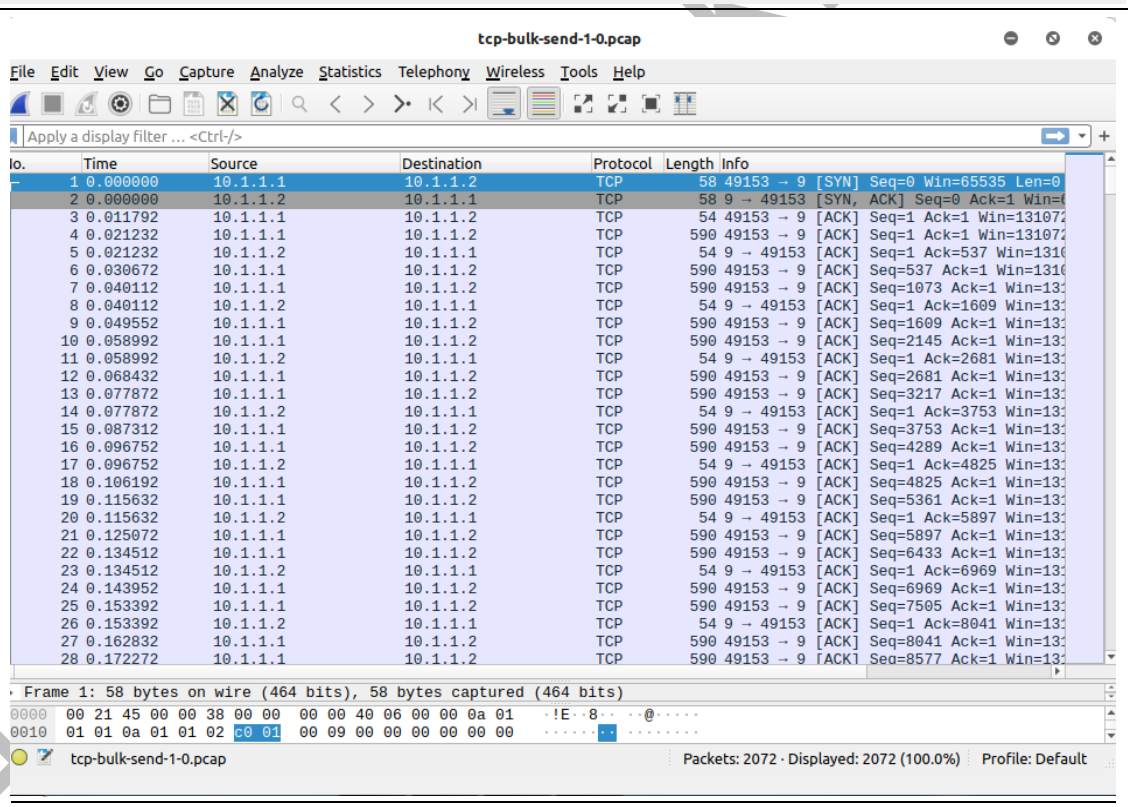

python
2
4
6
8
10
12
14
Zoom: 8.519
Speed: 1.000
Time: 0.000000 s
Snapshot
Simulate (F3)
Advanced


python
2
4
6
8
10
12
14
Zoom: 8.519
Speed: 1.000
Time: 4.500000 s
Snapshot
Simulate (F3)
Advanced


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
TCP:[ACK] S=91121 A=1
40.0,25.0
0
1


Cancel
Open
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
netanim-3.108
VBox_GA...
Other Locations
vaish
ns-allinone-3.32
ns-3.32
Name
Size
Type
Modified
src 7 Oct 2020
utils 7 Oct 2020
contrib 7 Oct 2020
examples 7 Oct 2020
bindings 7 Oct 2020
doc 7 Oct 2020
tcp-bulk-send-1-0.pcap 845.5 kB Packet Capture (PCAP) 16:17
tcp-bulk-send-0-0.pcap 845.5 kB Packet Capture (PCAP) 16:17
tcp-bulk-send.tr 2.6 MB Text 16:17
narender_ftp.xml 917.5 kB Markup 16:17
ftp-tcpp.xml 1.1 MB Markup Yesterday
tcp-large-transfer-2-0.pcap 2.5 MB Packet Capture (PCAP) Yesterday
tcp-large-transfer-1-1.pcap 2.5 MB Packet Capture (PCAP) Yesterday
tcp-large-transfer-1-0.pcap 2.5 MB Packet Capture (PCAP) Yesterday
All Files





CONCLUSION:

From this practical, I have learned about ftp bulk transfer in ns3.

AIM: ANALYZE THE NETWORK TRAFFIC AND PERFORMANCE PARAMETERS OF NETWORK USING WIRESHARK.

THEORY:

Wireshark:

Wireshark is a network or protocol analyser (also known as a network sniffer) available for free at the Wireshark website. It is used to analyze the structure of different network protocols and has the ability to demonstrate encapsulation. The analyser operates on Unix, Linux and Microsoft Windows operating systems, and employs the GTK+ widget toolkit and pcap for packet capturing. Wireshark and other terminal-based free software versions like Tshark are released under the GNU General Public License.

What Does Wireshark Mean?

- ➢ Wireshark is a free and open-source network protocol analyser that enables users to interactively browse the data traffic on a computer network. The development project was started under the name Ethereal, but was renamed Wireshark in 2006.
- ➢ Many networking developers from all around the world have contributed to this project with network analysis, troubleshooting, software development and communication protocols. Wireshark is used in many educational institutions and other industrial sectors.
- ➢ Wireshark shares many characteristics with tcpdump. The difference is that it supports a graphical user interface (GUI) and has information filtering features. In addition, Wireshark permits the user to see all the traffic being passed over the network.
- ➢ Features of Wireshark include:
  - Data is analyzed either from the wire over the network connection or from data files that have already captured data packets.
  - Supports live data reading and analysis for a wide range of networks (including Ethernet, IEEE 802.11, point-to-point Protocol (PPP) and loopback).
  - With the help of GUI or other versions, users can browse captured data networks.
  - For programmatically editing and converting the captured files to the editcap application, users can use command line switches.
  - Display filters are used to filter and organize the data display.
  - New protocols can be scrutinized by creating plug-ins.

- Captured traffic can also trace Voice over Internet (VoIP) calls over the network.
- When using Linux, it is also possible to capture raw USB traffic.

What Does Network Traffic Mean?

Network traffic refers to the amount of data moving across a network at a given point of time. Network data is mostly encapsulated in network packets, which provide the load in the network. Network traffic is the main component for network traffic measurement, network traffic control and simulation. The proper organization of network traffic helps in ensuring the quality of service in a given network.

Network traffic is also known as data traffic.

Network Traffic:

➢ Network traffic is the main component for bandwidth measurement and management. Moreover, various topologies of the network can only be implemented based on the amount of network traffic in the system.

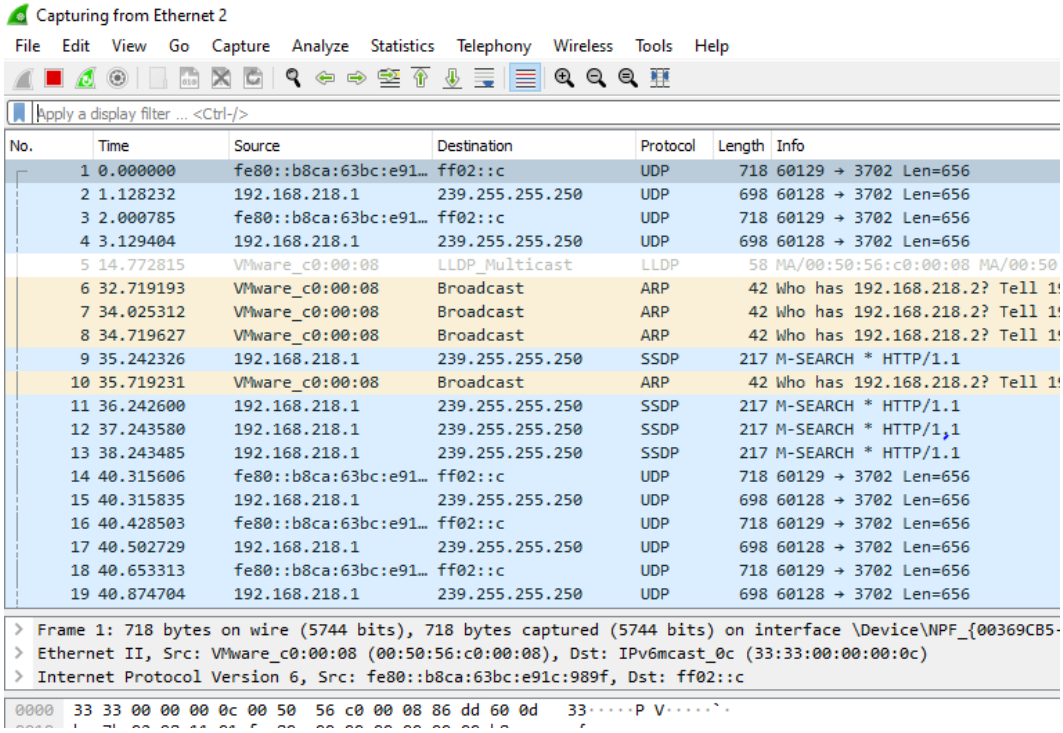
➢ Network traffic can be broadly classified into the following categories:

- Busy/heavy traffic - High bandwidth is consumed in this traffic
- Non-real-time traffic - Consumption of bandwidth during working hours
- Interactive traffic - Is subject to competition for bandwidth and could result in poor response times if prioritization of applications and traffic is not set
- Latency-sensitive traffic - Is subject to competition for bandwidth and could result in poor response times

➢ Proper analysis of network traffic provides the organization with the following benefits:

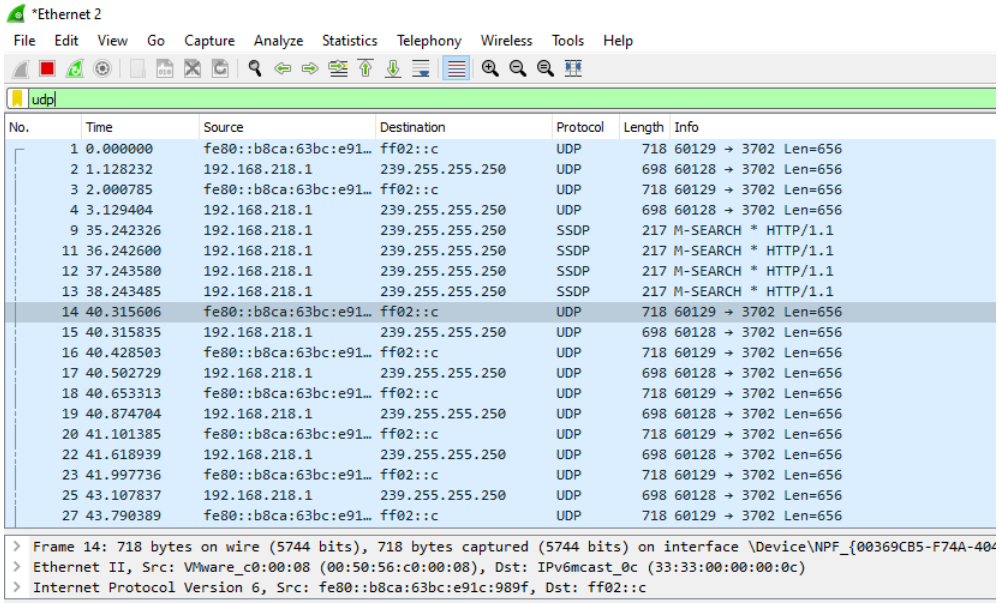
- Identifying network bottlenecks - There could be users or applications that consume high amounts of bandwidth, thus constituting a major part of the network traffic. Different solutions can be implemented to tackle these.
- Network security - unusual amount of traffic in a network is a possible sign of an attack. Network traffic reports provide valuable insights into preventing such attacks.
- Network engineering - Knowing the usage levels of the network allows future requirements to be analyzed.

START CAPTURING:

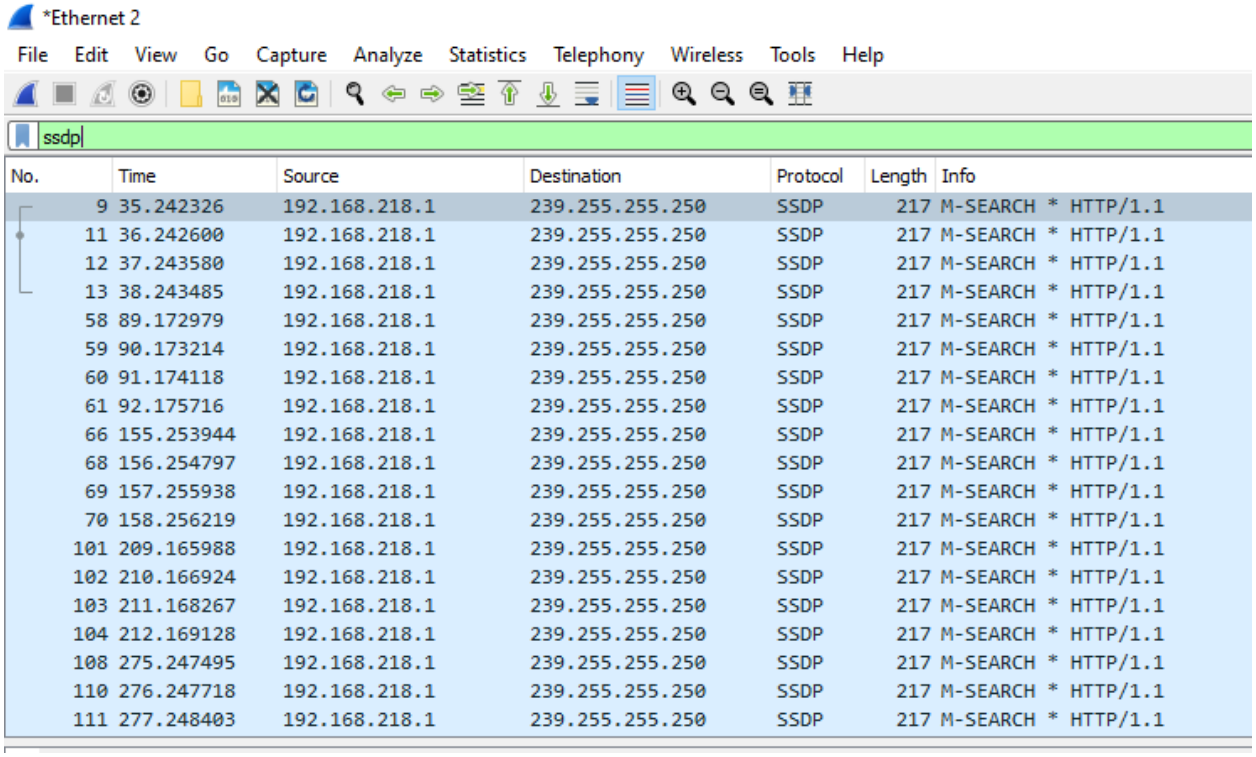


FILTER:

UDP filter:

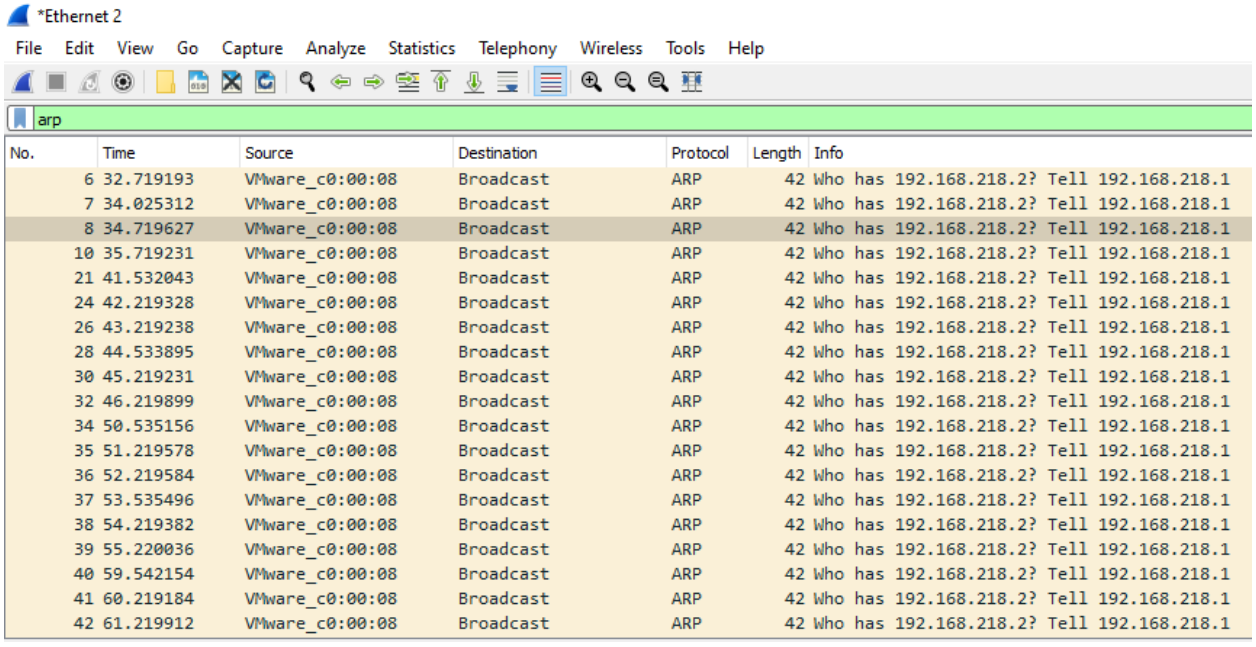


SSDP FILTER:



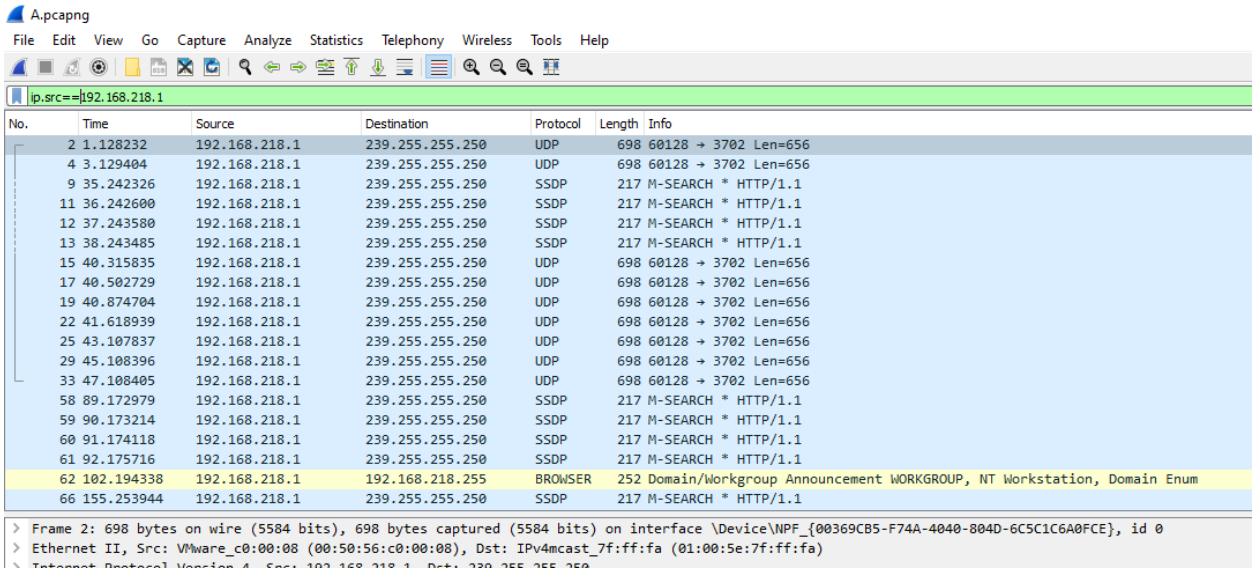
No.	Time	Source	Destination	Protocol	Length	Info
9	35.242326	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
11	36.242600	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
12	37.243580	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
13	38.243485	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
58	89.172979	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
59	90.173214	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
60	91.174118	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
61	92.175716	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
66	155.253944	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
68	156.254797	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
69	157.255938	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
70	158.256219	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
101	209.165988	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
102	210.166924	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
103	211.168267	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
104	212.169128	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
108	275.247495	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
110	276.247718	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
111	277.248403	192.168.218.1	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1

ARP FILTER:



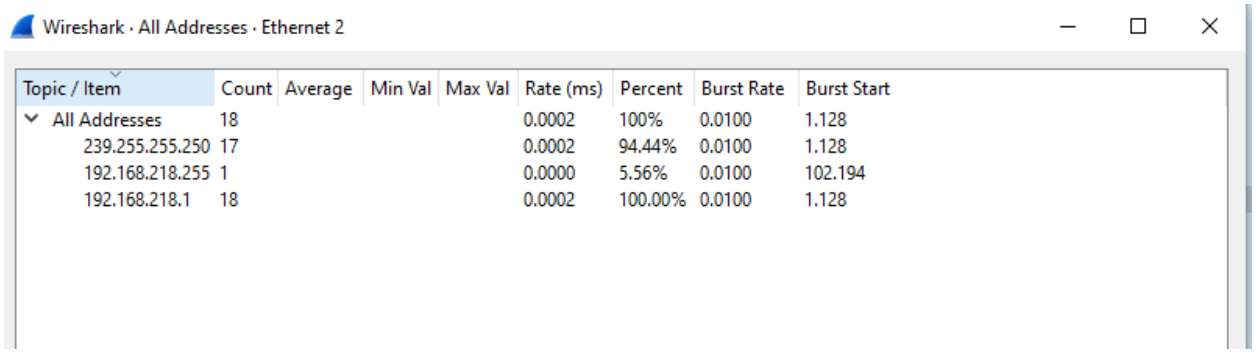
No.	Time	Source	Destination	Protocol	Length	Info
6	32.719193	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
7	34.025312	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
8	34.719627	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
10	35.719231	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
21	41.532043	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
24	42.219328	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
26	43.219238	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
28	44.533895	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
30	45.219231	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
32	46.219899	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
34	50.535156	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
35	51.219578	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
36	52.219584	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
37	53.535496	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
38	54.219382	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
39	55.220036	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
40	59.542154	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
41	60.219184	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1
42	61.219912	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.218.2? Tell 192.168.218.1

IP FILTER:

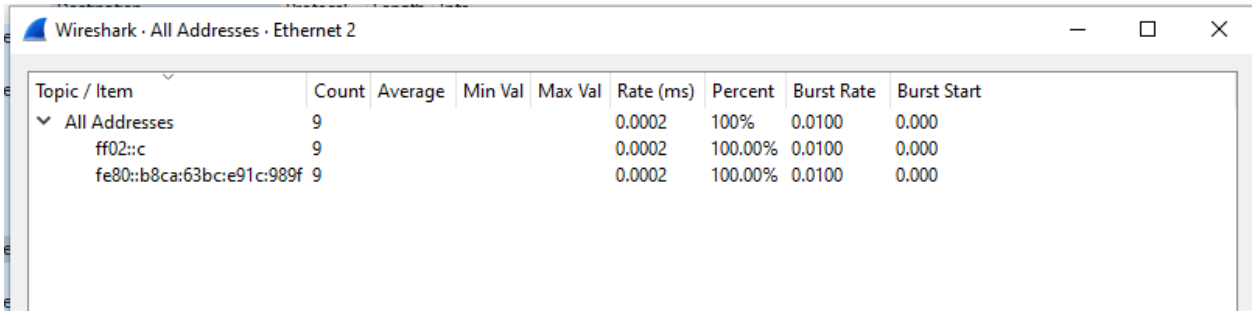


STATS:

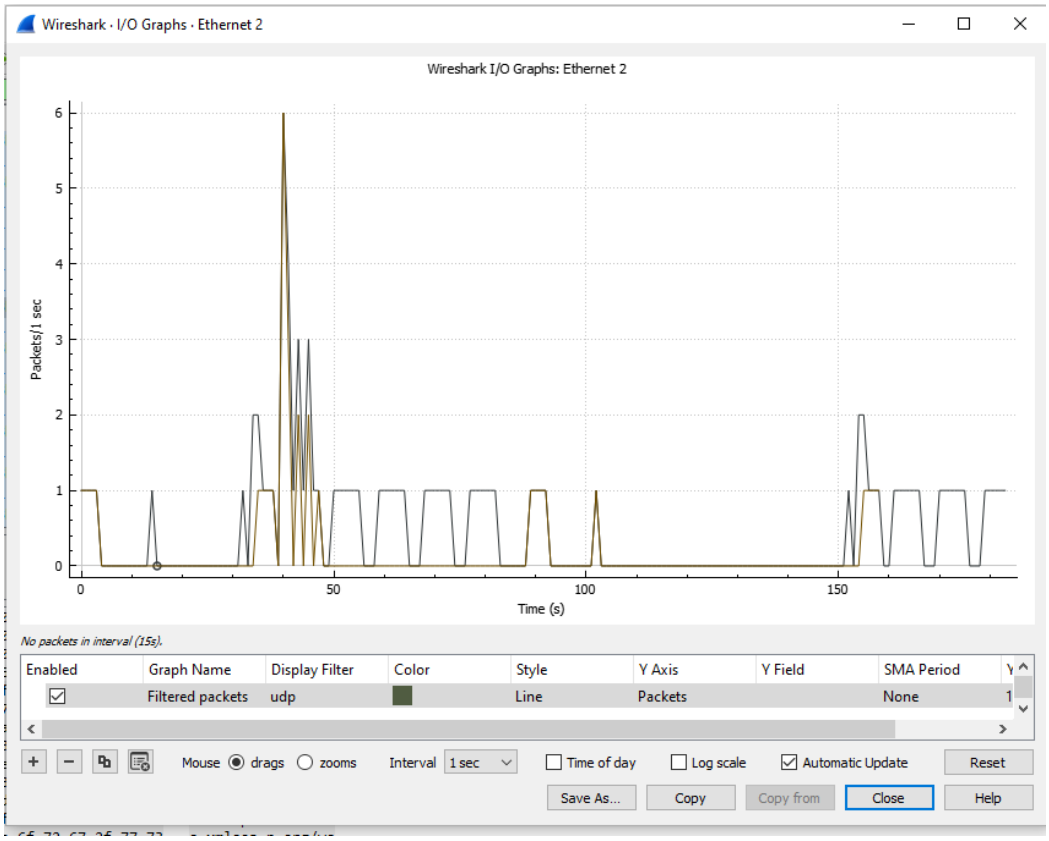
IPV4:



IPV6:



I/O GRAPH:



CONCLUSION:

Analyze the network traffic and performance parameters of the network using Wireshark executed successfully.