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 ();

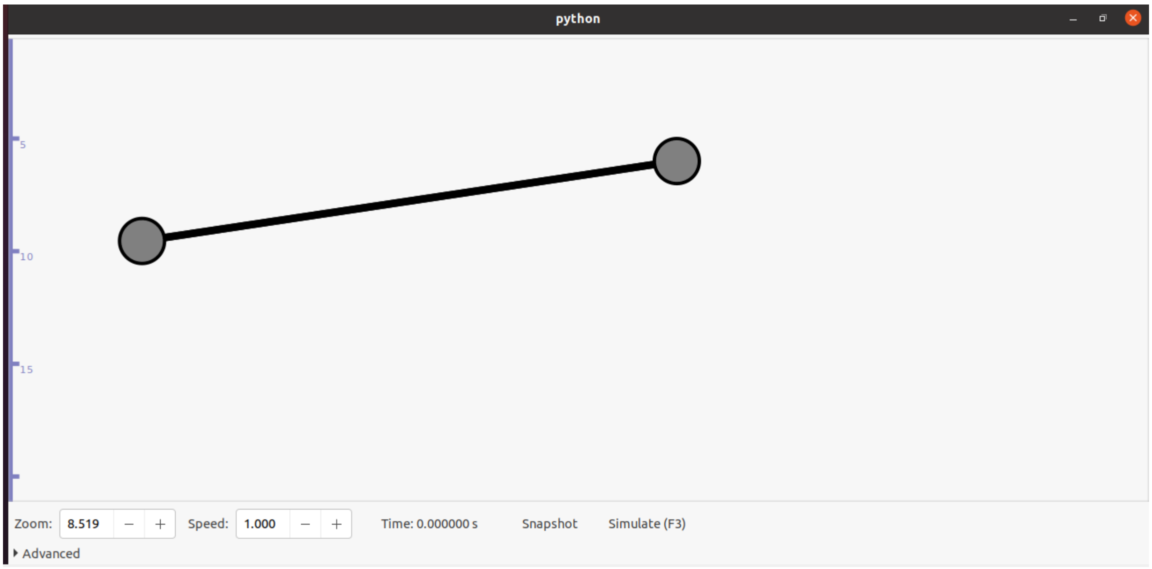
Simulator::Destroy ();

return 0;

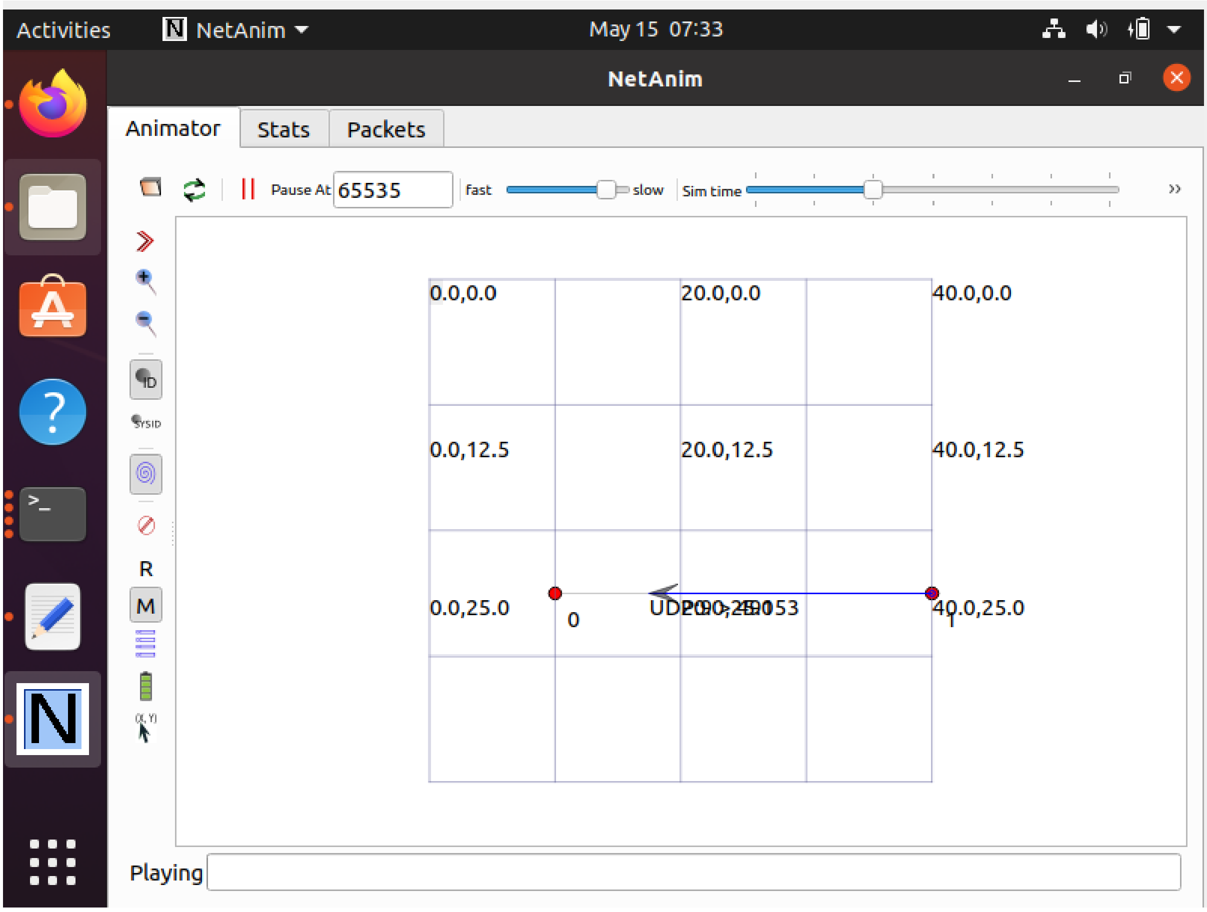
}

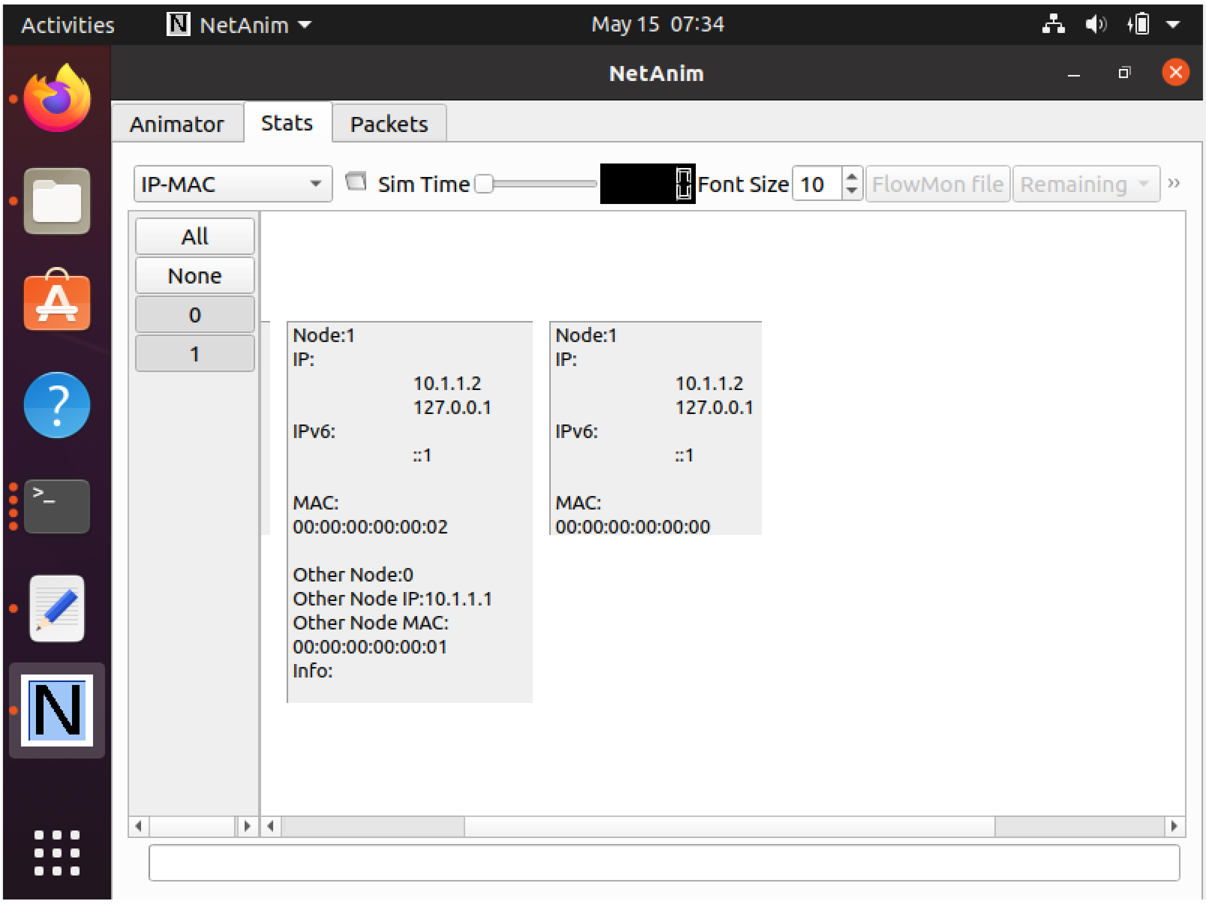
OUTPUT:

```
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$
```

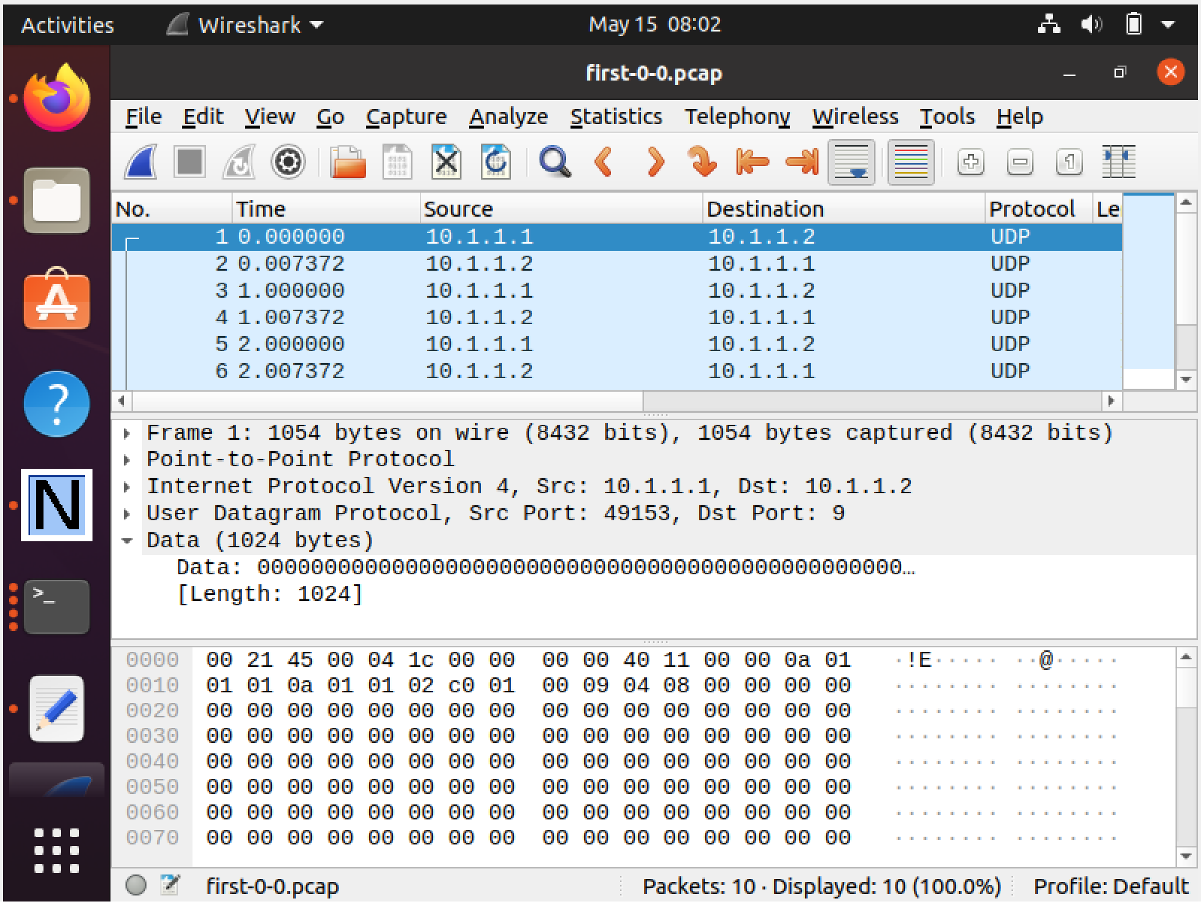

python
Zoom: 8.519
Speed: 1.000
Time: 0.000000 s
Snapshot
Simulate (F3)
Advanced

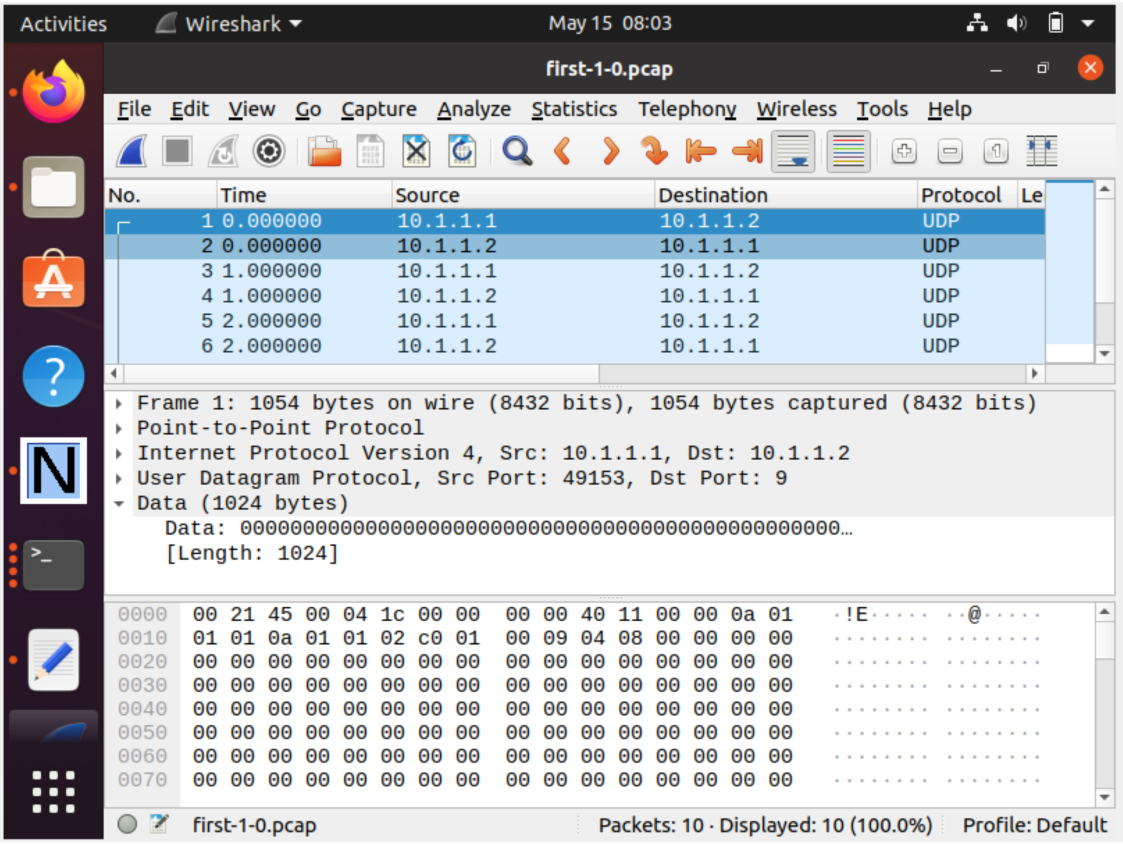
NetAnim:


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
Playing

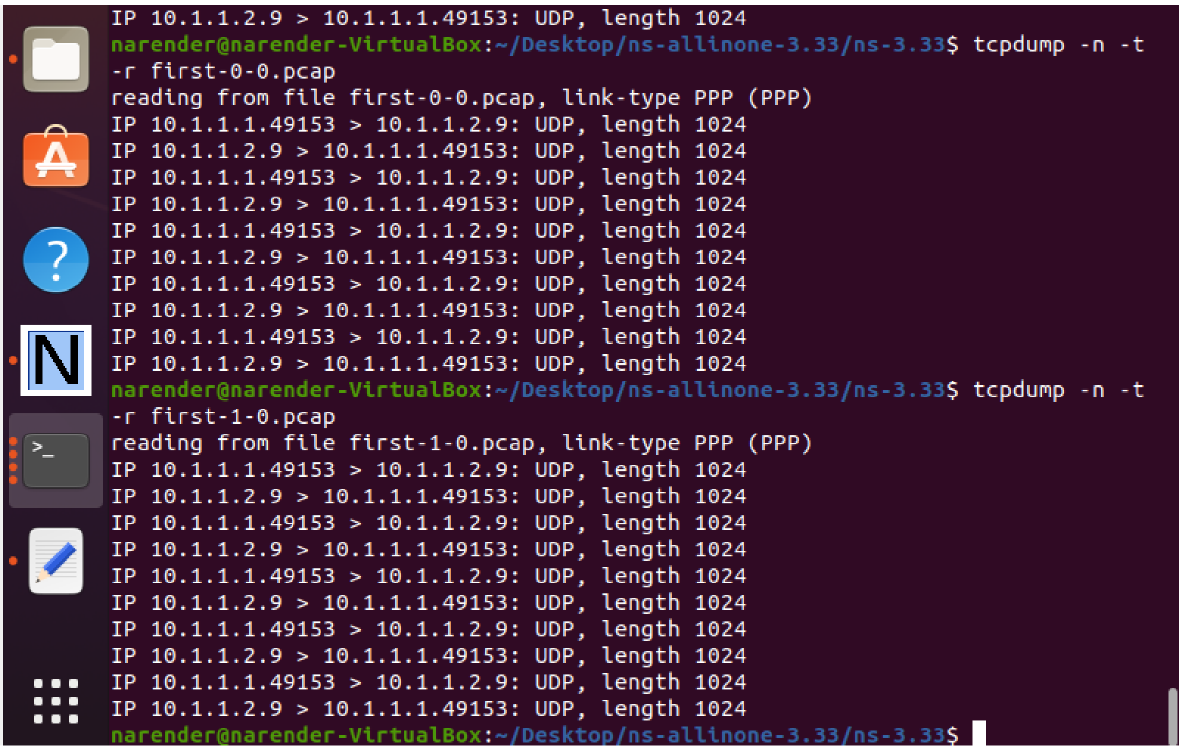


Packet Capture Trace:





Using TcpDump:



CONCLUSION:

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