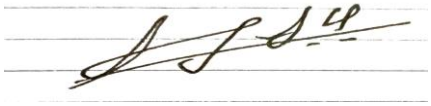
Name of Student: Sunny Satish Halkatti		
Roll Number: 17 (B)		**Lab Assignment Number: 2**
Title of Lab Assignment: Program to simulate traffic between two nodes		
DOP: 23/03/2023		**DOS: 13/04/2023**
CO: CO2	**PO:** PO1, PO2, PO3, P05,P07, PSO1	**Signature:** 

NWL - Practical 2

Aim: Program to simulate traffic between two nodes.

Theory:

Node.js In Internet jargon, a computing device that connects to a network is called a host or sometimes an end system. Because ns-3 is a network simulator, not specifically an Internet simulator, we intentionally do not use the term host since it is closely associated with the Internet and its protocols. Instead, we use a more generic term also used by other simulators that originates in Graph Theory — the node.
In 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. You should think of a Node as a computer to which you will add functionality. One adds things like applications, protocol stacks and peripheral cards with their associated drivers to enable the computer to do useful work. We use the same basic model in ns-3.
The definition of a node depends on the network and protocol layer referred to. A physical network node is an electronic device that is attached to a network, and is capable of creating, receiving, or transmitting information over a communication channel.

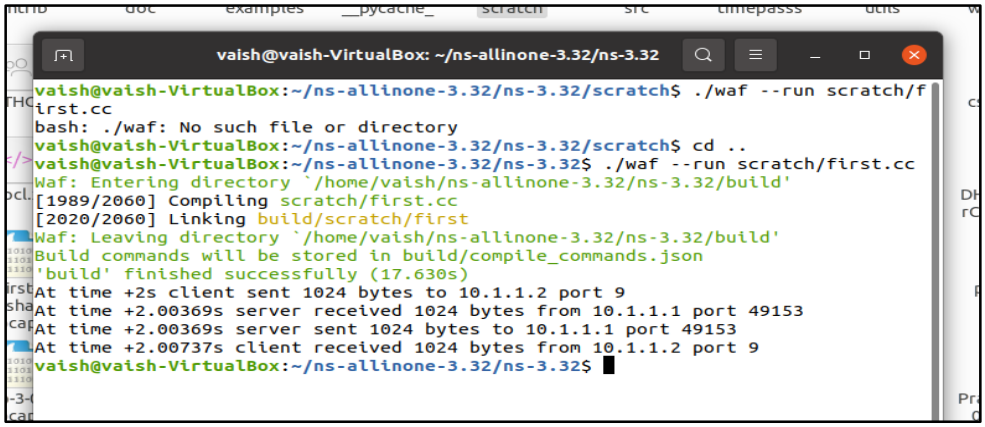
NetDevice- It used to be the case that if you wanted to connect a computer to a network, you had to buy a specific kind of network cable and a hardware device called (in PC terminology) a peripheral card that needed to be installed in your computer.
If the peripheral card implemented some networking function, they were called
Network Interface Cards, or NICs. Today most computers come with the network
interface hardware built in and users don't see these building blocks. A NIC will not work without a software driver to control the hardware. In Unix (or Linux), a piece of peripheral hardware is classified as a device. Devices are controlled using device drivers, and network devices (NICs) are controlled using network device drivers collectively known as net devices. In Unix and Linux you refer to these net devices by names such as eth0.
In ns-3 the net device abstraction covers both the software driver and the simulated hardware. A net device is "installed" in a Node in order to enable the Node to communicate with other Nodes in the simulation via Channels. Just as in a real computer, a Node may be connected to more than one Channel via multiple NetDevices. The net device abstraction is represented in C++ by the class NetDevice. The NetDevice class provides methods for managing connections to Node and Channel objects; and may be specialized by developers in the object-oriented programming sense. We will use the several specialized versions of the NetDevice called CsmaNetDevice, PointToPointNetDevice, and WifiNetDevice in this tutorial. Just as an Ethernet NIC is designed to work with an Ethernet network, the CsmaNetDevice is designed to work with a CsmaChannel; the PointToPointNetDevice is designed to work with a PointToPointChannel and a **WifiNetDevice** is designed to work with a WifiChannel.

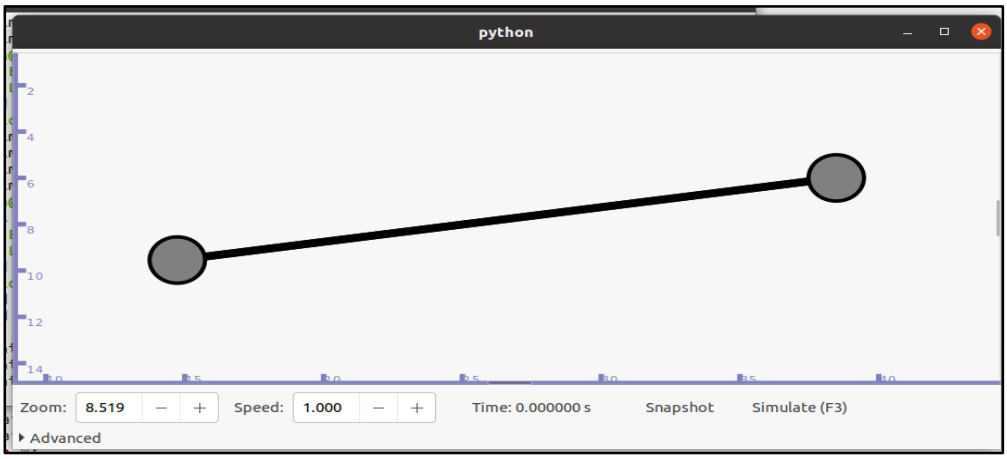
Code:

```
#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"

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");

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

  Time::SetResolution (Time::NS);
  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);

  NodeContainer nodes;
  nodes.Create (2);

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

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

  InternetStackHelper stack;
  stack.Install (nodes);

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

  Ipv4InterfaceContainer interfaces = address.Assign (devices);

  UdpEchoServerHelper echoServer (9);

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

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

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

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

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

Output:





Conclusion:

Simulation of traffic between two nodes(point to point) done successfully.