EXPERIMENT2

INTRODUCTION TO NETWORK SIMULATOR VERSION-3 (NS-3)

OBJECTIVE:

- Build and analyze simple topology using point to point link

BACKGROUND:

This is an introductory lab on network simulator version 3. We'll review some terms that are commonly used in networking, but have a specific meaning in ns-3. To explain the concepts and abstractions a code is also provided.

In ns-3 there is no concept of the operating system, the basic level of abstraction in ns-3 is the application level-user program. This abstraction is represented in C++ by the class named 'Application'. This class provides methods for managing the representation of developer's version of user-level application in simulations.

The basic ns-3 computing device abstraction is called node represented in C++ by the class 'Node'. Just as software applications run on computers to perform tasks in the "real world," ns-3 applications run on ns-3 Nodes to drive simulations in the simulated world.

Media over which the data flow in networks are called channels. In ns-3 one connects Node to an object representing a channel. Basic communication sub network abstraction is called the channel represented in C++ by the class 'Channel'. A simple channel can model a wire whereas specialized channels can model complicated things such as an Ethernet switch. Specialized versions of Channel that will be used in the lab are 'CSMA Channel', 'PointToPoint Channel' and 'Wi-Fi Channel'.

Network interface cards are controlled using the network device drivers called net devices referred in UNIX by names as eth0. A net device is installed in the Node in order to enable the Node to communicate with other Nodes in simulation via Channels. The netdevice abstraction represented in C++ by class 'NetDevices'. Several specialized netdevices are 'CsmaNetDevice', 'PointToPointNetDevice' and 'WifiNetDevice'. A Node may be connected to more than on Channel via multiple NetDevices.

In a large simulation network we need to connect many Nodes, NetDevices and Channels. Since connecting NetDevices to Nodes, NetDevices to Channels, assigning IP addresses, etc., are such common tasks in ns-3, we provide what we call 'topology helpers' to make this as easy as possible.

Visit the following link in order to understand the abstractions used in ns-3. It also provides you with the line by line explanation of the code given below:
https://www.nsnam.org/docs/release/3.40/tutorial/html/conceptual-overview.html#a-first-ns-3-scripT

Exercise1:

Understand each and every line of the above code, run the above script and comment on the output.

1. Modify the code so that 4 packets are sent and echoed.
2. Create 2 node with a single interface as shown in figure 2.1:
3. Point to point link
4. Data rate set to 8 Mbps, transmission delay set to 4ms
5. IP set to 192.168.40.0
6. Udp Echo Server Port set to 93
7. Packet size set to 512 bytes

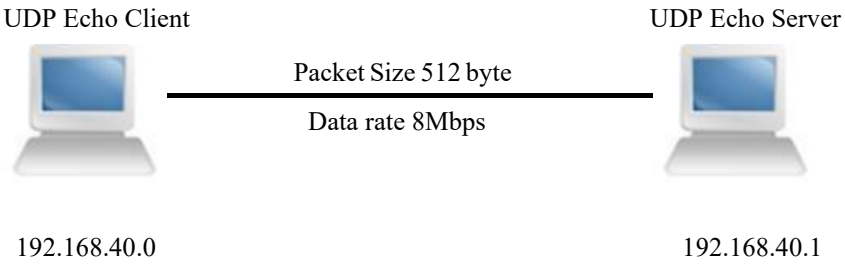


Figure 2.1: Point to Point Network

Paste your code in the space provided below.

```
#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[])
{
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 ("8Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("4ms"));

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

InternetStackHelper stack;
stack.Install (nodes);

Ipv4AddressHelper address;
address.SetBase ("192.168.40.0" , "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign (devices);

UdpEchoServerHelper echoServer (93);

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

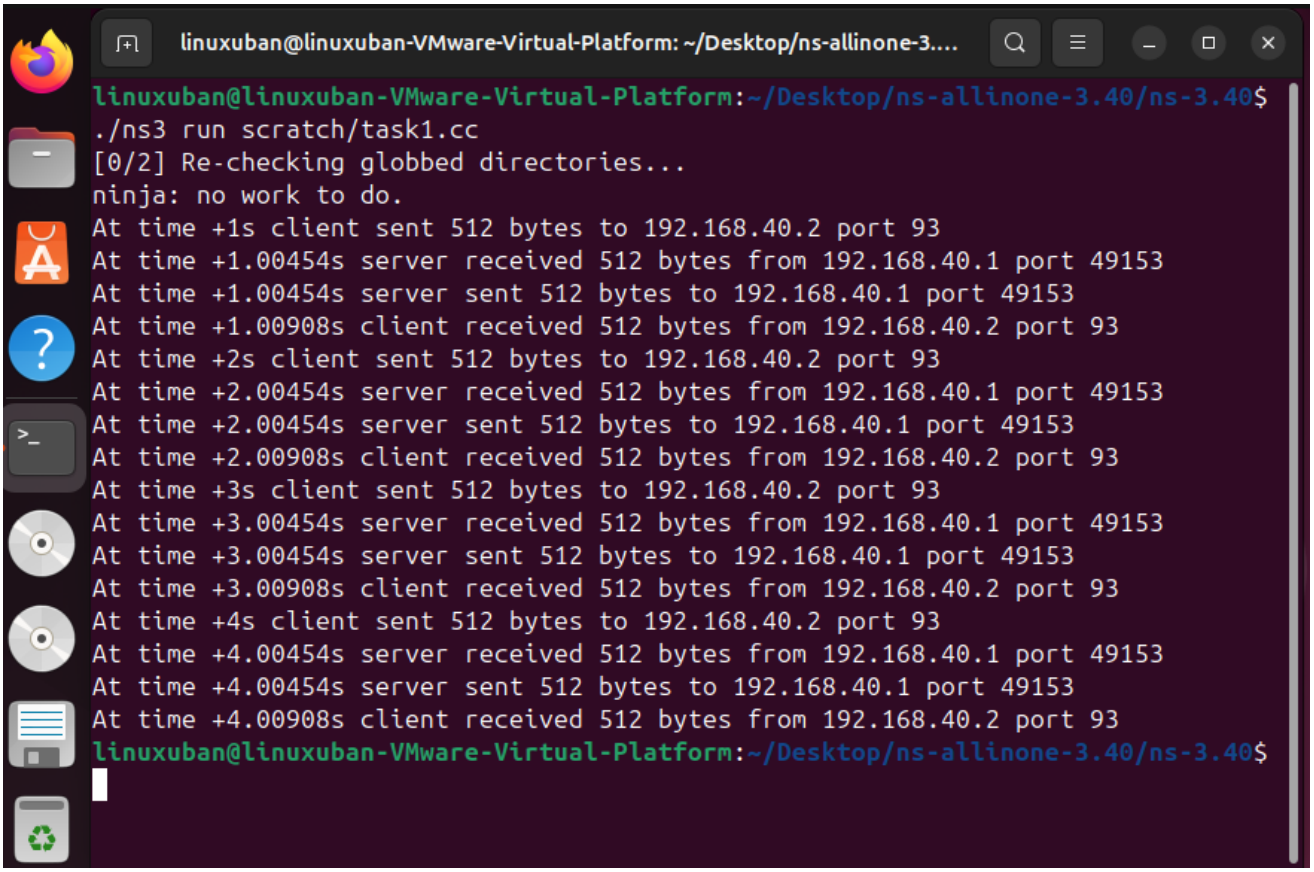
```
 UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 93);
 echoClient.SetAttribute ("MaxPackets", UintegerValue (4));
 echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
 echoClient.SetAttribute ("PacketSize", UintegerValue (512));

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

 clientApps.Start (Seconds (1.0));
 clientApps.Stop (Seconds (10.0));

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

Output:



Exercise 2:

Implement the network topology given in figure 2.2, both the outer nodes, node 1 and node 3, must send number of packets to the center node (node 2) at 5 and 10 seconds respectively. You must set the following parameters.

1. IP address of Node 1 and Node 3 should be 192.168.1.1 and 192.168.5.2 respectively.
2. Point to point link between Node 1 and Node 2
    a. Data rate set to 8 Mbps, transmission delay set to 8ms
3. Point to point link between Node 2 and Node 3
    a. Data rate set to 8 Mbps, transmission delay set to 4ms

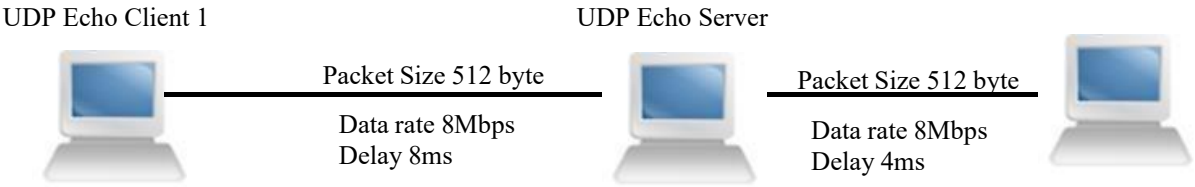


Figure 2.2

Paste your code in the space provided below.

```
 #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[]) {

  // Set the time resolution to nanoseconds

  Time::SetResolution(Time::NS);


  // Enable logging for the UdpEchoClientApplication and UdpEchoServerApplication at the INFO level

  LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);

  LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);



  // Create a container for network nodes

  NodeContainer nodes;

  nodes.Create(3);
```

```
// Create a point-to-point link between nodes 0 and 1

PointToPointHelper pointToPoint1;

pointToPoint1.SetDeviceAttribute("DataRate", StringValue("8Mbps"));

pointToPoint1.SetChannelAttribute("Delay", StringValue("8ms"));

NetDeviceContainer devices1;

devices1 = pointToPoint1.Install(nodes.Get(0), nodes.Get(1));


// Create another point-to-point link between nodes 1 and 2

PointToPointHelper pointToPoint2;

pointToPoint2.SetDeviceAttribute("DataRate", StringValue("8Mbps"));

pointToPoint2.SetChannelAttribute("Delay", StringValue("4ms"));

NetDeviceContainer devices2;

devices2 = pointToPoint2.Install(nodes.Get(1), nodes.Get(2));


// Install Internet stack on all nodes

InternetStackHelper stack;

stack.Install(nodes);


// Assign IPv4 addresses to the devices on the first link

Ipv4AddressHelper address1;

address1.SetBase("192.168.1.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces1 = address1.Assign(devices1);


// Assign IPv4 addresses to the devices on the second link

Ipv4AddressHelper address2;

address2.SetBase("192.168.5.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces2 = address2.Assign(devices2);


// Create a UDP echo server on node 1 listening on port 93

UdpEchoServerHelper echoServer(93);


// Install the UDP echo server application on node 1
```

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

// Start the server at time 1.0 seconds and stop it at time 20.0 seconds

serverApps.Start(Seconds(1.0));

serverApps.Stop(Seconds(20.0));


// Create a UDP echo client on node 0 to send packets to the server on node 1

UdpEchoClientHelper echoClient1(interfaces1.GetAddress(1), 93);

echoClient1.SetAttribute("MaxPackets", UintegerValue(1));

echoClient1.SetAttribute("Interval", TimeValue(Seconds(1.0)));

echoClient1.SetAttribute("PacketSize", UintegerValue(512));


// Install the UDP echo client application on node 0

ApplicationContainer clientApps1 = echoClient1.Install(nodes.Get(0));


// Start the client at time 5.0 seconds and stop it at time 10.0 seconds

clientApps1.Start(Seconds(5.0));

clientApps1.Stop(Seconds(10.0));


// Create another UDP echo client on node 2 to send packets to the server on node 1

UdpEchoClientHelper echoClient2(interfaces2.GetAddress(0), 93);

echoClient2.SetAttribute("MaxPackets", UintegerValue(1));

echoClient2.SetAttribute("Interval", TimeValue(Seconds(1.0)));

echoClient2.SetAttribute("PacketSize", UintegerValue(512));


// Install the second UDP echo client application on node 2

ApplicationContainer clientApps2 = echoClient2.Install(nodes.Get(2));


// Start the second client at time 11.0 seconds and stop it at time 15.0 seconds

clientApps2.Start(Seconds(11.0));

clientApps2.Stop(Seconds(15.0));
```

```
  Simulator::Run();


  Simulator::Destroy();

  return 0;
}
```

Output: