Describe each step and how to run your program :

First step, declare the “**NodeContainer**” variable called “nodes” and use “**Create()**” to create three nodes(**node 0 for client, node 1 for server 1 and node 2 for server 2**).

```
NodeContainer nodes;
nodes.Create(3);
```

Second step, declare the “**PointToPointHelper**” and “**NetDeviceContainer**” variables. We get two variable per data structure, 1 for flow 1(node 0 to node 1), 2 for flow 2(node 0 to node2).
Use “**SetDeviceAttribute()**” to set the data rate and use “**SetChannelAttribute()**” to set the delay. Once the attributes are set, we can create the devices. Install them on the required nodes and connect the devices together using a PointToPoint channel by using “**Install()**”.

```
PointToPointHelper pointToPoint1, pointToPoint2;
NetDeviceContainer devices1, devices2;
pointToPoint1.SetDeviceAttribute("DataRate", StringValue("2Mbps"));
pointToPoint1.SetChannelAttribute("Delay", StringValue("2ms"));
devices1 = pointToPoint1.Install(nodes.Get(0), nodes.Get(1));
pointToPoint2.SetDeviceAttribute("DataRate", StringValue("3Mbps"));
pointToPoint2.SetChannelAttribute("Delay", StringValue("2ms"));
devices2 = pointToPoint2.Install(nodes.Get(0), nodes.Get(2));
```

Third step, declare the “**InternetStackHelper”** variable called “stack”.
Use “**Install()**” to aggregate implementations of the Ip, Udp, and Tcp classes for each node in the “nodes”.

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

Fourth step, declare the “**Ipv4AddressHelper”** variables called “address1” and “address2”.
Use “**SetBase()**” to set the **IPv4** and **subnet mask**.

```
Ipv4AddressHelper address1, address2;
address1.SetBase("10.0.1.0", "255.255.255.0");
address2.SetBase("10.0.2.0", "255.255.255.0");
```

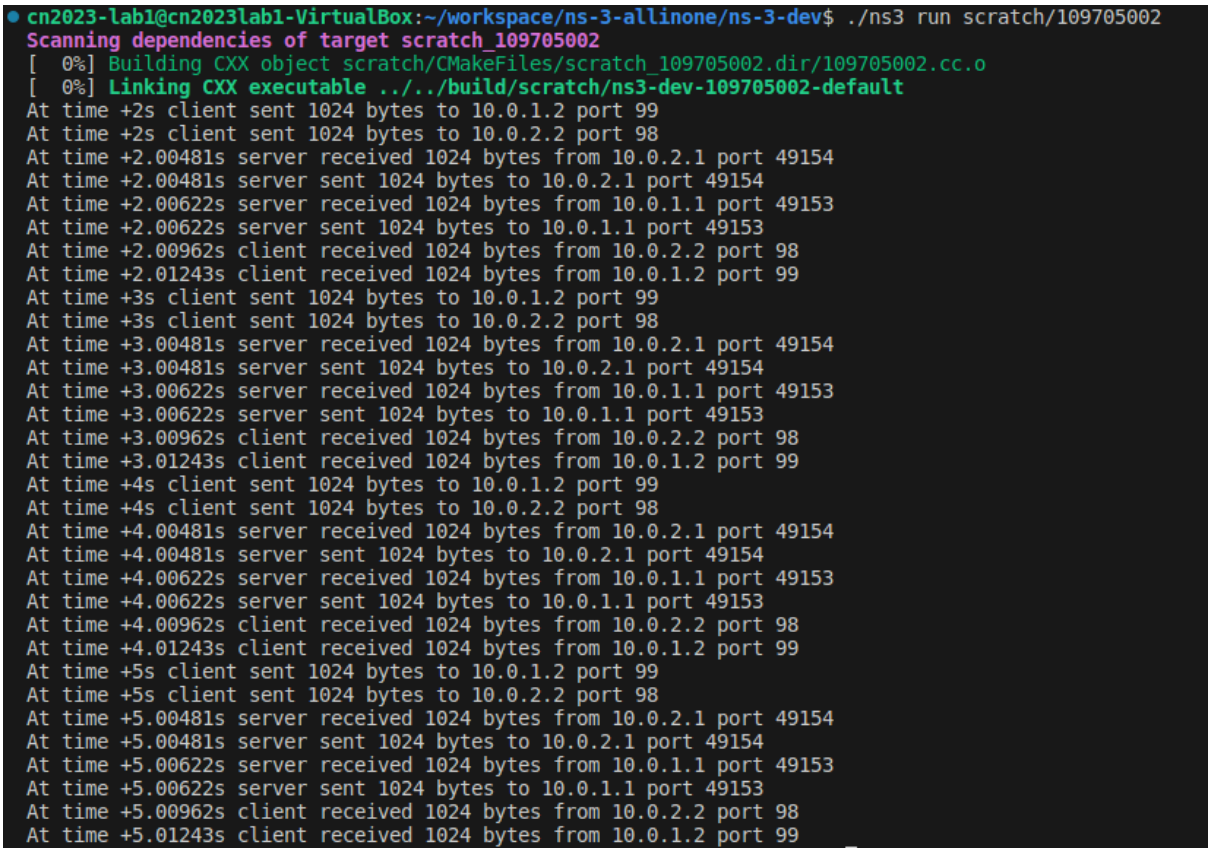
Fifth step, declare the “**Ipv4InterfaceContainer”** variables called “interface1” and “interface2”.
The interfaces are created by “**Assign()**” which performing the actual address assignment to the devices.

```
Ipv4InterfaceContainer interfaces1 = address1.Assign(devices1);
Ipv4InterfaceContainer interfaces2 = address2.Assign(devices2);
```

Last but not least, declare the "**UdpEchoServerHelper"** variables and provide them with port numbers due to listen on ports 99 and 98.
Therefore, use **"Install()**" to install the UDP echo servers on nodes in the simulation through "**ApplicationContainer**" variables.
By "**Start()**" and "**Stop()**", we can set the starting points and ending points of the sending flows.
Then, declare the "**echoClientHelper**" variables to send UDP packets to the echo servers on ports 99 and 98 and set the attributes like maximum number of the packets, interval and sizes of packet.
After setting up two flows, just use "**Simulator::Run()**" to simulate and "**Simulator::Destroy()**" to end the simulation.

```
UdpEchoServerHelper echoServer1(99), echoServer2(98);

ApplicationContainer serverApps1 = echoServer1.Install(nodes.Get(1));
serverApps1.Start(Seconds(1.0));
serverApps1.Stop(Seconds(10.0));

ApplicationContainer serverApps2 = echoServer2.Install(nodes.Get(2));
serverApps2.Start(Seconds(1.0));
serverApps2.Stop(Seconds(10.0));

UdpEchoClientHelper echoClient1(interfaces1.GetAddress(1), 99);
echoClient1.SetAttribute("MaxPackets", UintegerValue(4));
echoClient1.SetAttribute("Interval", TimeValue(Seconds(1.0)));
echoClient1.SetAttribute("PacketSize", UintegerValue(1024));

ApplicationContainer clientApps1 = echoClient1.Install(nodes.Get(0));
clientApps1.Start(Seconds(2.0));
clientApps1.Stop(Seconds(10.0));

UdpEchoClientHelper echoClient2(interfaces2.GetAddress(1), 98);
echoClient2.SetAttribute("MaxPackets", UintegerValue(4));
echoClient2.SetAttribute("Interval", TimeValue(Seconds(1.0)));
echoClient2.SetAttribute("PacketSize", UintegerValue(1024));

ApplicationContainer clientApps2 = echoClient2.Install(nodes.Get(0));
clientApps2.Start(Seconds(2.0));
clientApps2.Stop(Seconds(10.0));

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

To run the program, using “cd” to move to the folder “**ns-3-allinone/ns-3-dev**” and input the command “**./ns3 run scratch/109705002**”



Answer the following question in short:

• What is the different between network simulation and emulation?

Network simulation provides a controlled environment for testing by creating models, while network emulation provides a more realistic environment that closely resembles real-world conditions by mimic networks. However, network emulation is usually more complex and expensive.

• Generally, in NS-3, if you don’t change the code, the output will be always the same every time you run, even if you set some probabilistic parameter like error rate, why?

Network simulators like ns-3 aim to provide determinism and reproducibility in simulations. Making other people can replicate the same results by running the same simulation.

• Following the previous question, how to deal with this problem?

We can set Seed explicitly by add “**RngSeedManager::SetSeed()**” in codes and add “**--RngRun**” in command option.

Bonus

• What have you learned from this lab?

I have learned how to use ns-3 to simulate a network. I think it is important to obtain the skill of creating an SDN- at least need to know how to modify codes, which are related to an SDN, written by others.

• What difficulty have you met in this lab?

I think this lab is relatively hard to begin because there are too many data structures and functions that we do not know how to use. Also, even if we have a sample code(first.cc), we still need some time to figure out each part of the code. Yet, I still find some joy while I deal with the lab!