Experiment 2 - Introduction to Network Simulator Version-3 (NS-3)

Muhammad Atif Farooq L144392 EL-A

12th September 2019

1 Introduction

NS-3 is an open-source discrete-event network simulator for internet systems, built using C++ and Python. It is used primarily for research and educational purposes and is publicly available for use under the GNU GPLv2 license.

2 Objective

To build and analyze a simple point to point link topolgy in NS-3.

3 Procedure

Before discussing the procedure we must remark that the following experiment was performed on a linux machine using the Ubuntu 18.04 LTS distribution. We now discuss the expermiment in the following sections starting out with the installation and build.

3.1 Installing and Building NS-3

To Install NS-3 we proceed to the link `https://www.nsnam.org/releases` and download the latest release as a tarball compressed file. As of this writing `ns-3.30` is the latest release thus we use it as our reference. Upon a successful download extract the file into your preferred directory and enter the extracted `ns3-allinone` directory using your terminal. Finally execute `build.py` script using `./build.py --enable-examples --enable-tests` and wait until all required packages are installed.

3.2 Building and Executing Scripts

Having built NS-3, copy the `first.cc` file present in `examples/tutorials` and paste it in the the `scratch` directory, following this navigate back to the `ns-3.30`

directory and execute the `./waf --run scratch/first` command inside the `ns-3.30` directory to build and run the `first.cc` script.To run your own scripts write them inside the `scratch` directory and follow the remaining procedure as stated above. You should see the following output on your terminal, after executing `./waf --run scratch/first`.

```
atifcppprogrammer@pop-os:~/Desktop/Semester/DCN/Lab/ns-allinone-3.30/ns-3.30$ ./waf --run scratch/first
Waf: Entering directory `/home/atifcppprogrammer/Desktop/Semester/DCN/Lab/ns-allinone-3.30/ns-3.30/build'
Waf: Leaving directory `/home/atifcppprogrammer/Desktop/Semester/DCN/Lab/ns-allinone-3.30/ns-3.30/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.432s)
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
atifcppprogrammer@pop-os:~/Desktop/Semester/DCN/Lab/ns-allinone-3.30/ns-3.30$
```

Figure 1: first.cc

3.3 Task

As directed by the manual we first commented on the output of the `first.cc` script presented in Figure-1 which indicates that the client intiates the communication by sending a packet to the server, the server then receives the said packet, and then sends packet of its own to the client acknowledging that the original packet was successfully received by it.

The Lab task also required us to modify the above code, to change the packet size, data rate, transmisson delay, IP addressing of the nodes and the server port. The Code and its accompanying output are presented 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[])
{
  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 ("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","0.0.0.0");

  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 (1));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (512));

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

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

4 Conclusions

NS-3 is allows us to model and analyze a given network.

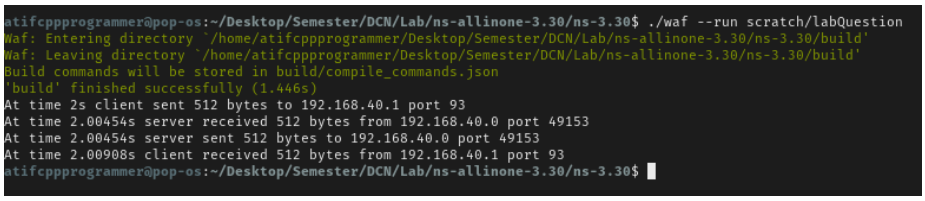

Figure 2: labQuestion.cc

5 Post-Lab Question

The Code and its accompanying output is presented 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 ("atifcppprogrammer_post_lab");

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);

  // Creating Node Container for Our Two Client Nodes and One Server Node.
  NodeContainer allNodes;
  allNodes.Create(3);

  // Node Container for first Client-Server Point to Point Link.
  NodeContainer nodesOne;
  nodesOne = NodeContainer(allNodes.Get(0),allNodes.Get(1));

  // Node Container for second Client-Server Point to Point Link.
  NodeContainer nodesTwo;
  nodesTwo = NodeContainer(allNodes.Get(1),allNodes.Get(2));
```

```
// Specifying PointToPoint Link for nodesOne Container.
PointToPointHelper pointToPointOne;
pointToPointOne.SetDeviceAttribute("DataRate",StringValue("8Mbps"));
pointToPointOne.SetChannelAttribute("Delay",StringValue("8ms"));

// Specifying PointToPoint Link for nodesTwo Container.
PointToPointHelper pointToPointTwo;
pointToPointOne.SetDeviceAttribute("DataRate",StringValue("4Mbps"));
pointToPointOne.SetChannelAttribute("Delay",StringValue("4ms"));

// Installing Network-Cards on nodesOne.
NetDeviceContainer devicesOne;
devicesOne = pointToPointOne.Install(nodesOne);

// Installing Network-Cards on nodesTwo.
NetDeviceContainer devicesTwo;
devicesTwo = pointToPointTwo.Install(nodesTwo);

// Installing Protocol Stack on First Client and Server;
InternetStackHelper stackOne;
stackOne.Install(nodesOne);

// Installing Protocol Stack on Second Client.
InternetStackHelper stackTwo;
stackTwo.Install(nodesTwo.Get(1));

// Specifying and Assigning IP-Addresses for First Client and Server Link.
Ipv4AddressHelper addressOne;
addressOne.SetBase ("192.168.1.0", "255.255.255.0","0.0.0.1");

Ipv4InterfaceContainer interfacesOne = addressOne.Assign (devicesOne);

// Specifying and Assigning IP-Addresses for Server and Second Client Link.
Ipv4AddressHelper addressTwo;
addressTwo.SetBase ("192.168.5.0", "255.255.255.0","0.0.0.1");

Ipv4InterfaceContainer interfacesTwo = addressTwo.Assign (devicesTwo);

// Installing UdpEchoServer on Central Node.
UdpEchoServerHelper echoServer (9);

ApplicationContainer serverApps = echoServer.Install (nodesOne.Get(1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (15.0));

// Configuring and Installing First Client App.
```

```
  UdpEchoClientHelper echoClientOne (interfacesOne.GetAddress (1), 9);
  echoClientOne.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClientOne.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClientOne.SetAttribute ("PacketSize", UintegerValue (512));

  ApplicationContainer clientAppsOne = echoClientOne.Install (nodesOne.Get (0));
  clientAppsOne.Start (Seconds (5.0));
  clientAppsOne.Stop (Seconds (15.0));

  // Configuring and Installing Second Client App.
  UdpEchoClientHelper echoClientTwo (interfacesTwo.GetAddress (0), 9);
  echoClientTwo.SetAttribute ("MaxPackets", UintegerValue (1));
  echoClientTwo.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClientOne.SetAttribute ("PacketSize", UintegerValue (512));

  ApplicationContainer clientAppsTwo = echoClientTwo.Install (nodesTwo.Get (1));
  clientAppsTwo.Start (Seconds (10.0));
  clientAppsTwo.Stop (Seconds (15.0));

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

```
atifcppprogrammer@pop-os:~/Desktop/Semester/DCN/Lab/ns-allinone-3.30/ns-3.30$ ./waf --run scratch/postLabQuestion
Waf: Entering directory `/home/atifcppprogrammer/Desktop/Semester/DCN/Lab/ns-allinone-3.30/ns-3.30/build'
Waf: Leaving directory `/home/atifcppprogrammer/Desktop/Semester/DCN/Lab/ns-allinone-3.30/ns-3.30/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.357s)
At time 5s client sent 512 bytes to 192.168.1.2 port 9
At time 5.00508s server received 512 bytes from 192.168.1.1 port 49153
At time 5.00508s server sent 512 bytes to 192.168.1.1 port 49153
At time 5.01017s client received 512 bytes from 192.168.1.2 port 9
At time 10s client sent 100 bytes to 192.168.5.1 port 9
At time 10.0317s server received 100 bytes from 192.168.5.2 port 49153
At time 10.0317s server sent 100 bytes to 192.168.5.2 port 49153
At time 10.0635s client received 100 bytes from 192.168.5.1 port 9
atifcppprogrammer@pop-os:~/Desktop/Semester/DCN/Lab/ns-allinone-3.30/ns-3.30$
```

Figure 3: postLabQuestion.cc