Department of Computer Science, Electrical and Space Engineering

ADVANCED WIRELESS NETWORKS

Lab0 Report – Introduction to NS3

Otabek Sobirov

6th September 2021

Table of Contents

Questions

1. **What are alternatives to installing NS3 under different operating systems?**

NS-3 has been developed and still being supported to run mainly on Linux-based and MacOS operating systems. If Window OS is desired, creating a virtual Linux machine with VirtualBox or VMware is recommended.

***Linux-based OS*:** NS3 can be installed in two different ways: manually or with Bake.

a. In the first way, NS3 software will be downloaded and unzipped(and **./build.py** should be run to compile all the libraries and source files).
b. The other way uses a software tool called "**Bake**". We'll just clone it to our machine using Git and follow "Bake" instructions.

The minimum requirements are the same for all the operating systems: **C++ compiler** (gcc/g++) and **Python** (version 3) interpreter.

***MacOS*:** For the "Cataline" version of MacOS, there is a tool called Xcode. Xcode has a feature "Xcode Command Line Tools" that is used to install NS3. Xcode can be downloaded from App Store.

2. **Write a step-by-step instruction for creating a simulation scenario**

The logical steps are as follows.

1. Creating communication nodes.
2. Choosing an appropriate communication protocol.
3. Creating network devices by adding the protocol to the nodes
4. Creating a network domain and defining the domain IP address.
5. Defining Server and Client types and creating their applications.
6. Running the simulation and collecting logs

The practical instructions to create a simulation scenario is given below. In this example, the script file of a simulation is written in **C++.**

1. Importing all the dependent libraries.

```
17  #include "ns3/core-module.h"
18  #include "ns3/network-module.h"
19  #include "ns3/internet-module.h"
20  #include "ns3/point-to-point-module.h"
21  #include "ns3/applications-module.h"
```

2. We will define the namespace as ns3. This helps to avoid name confusions of different namespaces. We also call CommandLine class to get some variable from the terminal during the execution

```
32  using namespace ns3;
33
34  NS_LOG_COMPONENT_DEFINE ("FirstScriptExample");
35
36  int
37  main (int argc, char *argv[])
38  {
39    CommandLine cmd (__FILE__);
40    cmd.Parse (argc, argv);
```

3. Setting the simulation time and enabling terminal logging.

```
44    Time::SetResolution (Time::NS);
45    LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
46    LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
```

4. Creating two network nodes.

```
50    NodeContainer nodes;
51    nodes.Create (2);
```

5. Creating a point-to-point communication channel and setting the bandwidth and delay rate of the channel.

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

6. Assigning the channel to the created nodes.

```
61    NetDeviceContainer devices;
62    devices = pointToPoint.Install (nodes);
```

7. Creating an IP address domain and assigning IP addresses to the created network devises.

```
71    Ipv4AddressHelper address;
72    address.SetBase ("10.1.1.0", "255.255.255.0");
73    Ipv4InterfaceContainer interfaces = address.Assign (devices);
```

8. Creating a UDP server and assigning a port number and an IP address, then staring it as an application.

```
77    UdpEchoServerHelper echoServer (9);
78
79    ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
80    serverApps.Start (Seconds (1.0));
81    serverApps.Stop (Seconds (10.0));
```

9. Creating a UDP client types and defining its attributes

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

10. Creating a client application using the UDP client type previously defined. Setting the starting and stopping time of the client application.

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

11. Starting the simulation and destroying after it finishes.

```
 97    Simulator::Run ();
 98    Simulator::Destroy ();
 99    return 0;
100  }
```

3. If you would need to simulate a protocol which is not inside the NS3 library, what would you need to do?

If a particular protocol is not in the NS3 module library, we can search it online. If there is any on the internet, we can place that module in ***src/*** folder. However, usually, when we install NS3 and build for the first time, all the available modules will be installed alongside.

If we cannot find a module (protocol) that we are searching, it is possible to create a module for that protocol and place it in ***src/*** folder. NS3 provides a folder structure to create a module.

4. For compiling an ns-3 executable, a special built-in system is used. What is this system?

Waf is a Python-based built-in tool in NS3. It compiles source files and place the executables of them in ***build/*** folder. It also manages dependencies while compiling the files.

5. Describe the purpose of different (only main) folders of the ns3 distribution

The main folders in NS3 software are bindings, build, examples, doc, scratch and src(source).

1. *bindings/* folder contains files that help to integrate Python into NS3.
2. In *build/* folder, all the compiled executables of scripts and libraries can be found. When we create a new script, waf compiles it into *build/* folder. And next time when we try to run the same script, it starts checking from *build/* folder.
3. *example/* folder contains implementation examples of different network simulations. These examples use built-in libraries in NS3.
4. *doc/* folder has the user manual, model documentations and NS3 tutorial files.
5. *scratch/* is the place in which users put their simulation script files.
6. In *src/* folder, we can find the source code for NS3 simulations.

6. Which folder should contain your simulation scripts?

User scripts should be placed in *scratch/* folder. If a user wants to create a new module, it should be placed in *src/* folder by using the module folder structure provided on the NS3 tutorials website.

7. Write a step-by step instruction for executing an ns-3 simulation

1. We create a simulation script file and place it into *scratch/* folder.
2. Go one folder back
3. Run the command: ./waf - -run *scratch/filename* (for .cc files)
4. Run the command: ./waf - -pyran *scratch/filename.py* (for .py files)

This command builds the executable file and runs it in the terminal. Next time when we run the same file, "waf" checks the *build/* folder for the executable file and runs from there. If not found, waf builds again and executes.

8. In how many formats does ns-3 save the results (traces) of a simulation? Name them. What are the major differences?

The trace files of a simulation are stored in two different file formats: **ASCII** (*filename.tr)* and **PCAP** *(filename.pcap*).

a. The first file format stores logs in plain text (for example, a node sent some size of data to the other node via an IP address).
b. The latter file format stores packet probes as logs. PCAP files can be analyzed using **Wireshark** software tool. PCAP files includes the headers of all the protocols used in the communication between the nodes.

For both of the file formats, users should explicitly write logging class objects in the script code to create these log files.

9. When you run your simulation in which folder will you find the simulation traces?

a. When we run a simulation (a new script), waf tool compiles it and puts the resulting executable and object files into the *build/* folder (corresponding path, in our case: *build/scratch/*).
b. Log files can be stored in two formats **ASCII** (*file_name.tr*) and **PCAP** (*file_name.pcap*). Log files are stored in the root folder of NS3. If we create an .**xml** file for animation, it is also stored in this folder.