Riga Technical University

RAE411 Telecommunication Software

Faculty of Telecommunications

Exercise 4

NS3 Simulator

Fall 2019

By
Matthew Duncan Weston, 190AEM020

November 19, 2019

Contents

1 Introduction

This rapport documents the experiences gathered so far with installing and learning how to simulate network protocols using the NS-3 simulation enviroment. Meanwhile a scientific paper on research of using Machine Learning tools together with the NS-3 has been presented for analysis. In the second part of this rapport the paper will be discussed.

2 NS3 Installation and Setup

Correct installation of the NS3 simulator software requires carefully following steps, which include updating the linux terminal, installation of neccesary libraries and correct working enviroment cofiguration. From several different approaches the following proved the most succesful and includes NetAnim, a useful visualization tool and a script editor application called Eclipse.

First any automatic updates were installed followed by a command which installs all neccesary libraries to ensure that the ns3 can be installed correctly. Among these are g++ and python libraries.

```
$ sudo apt-get update
```

```
$ sudo apt-get install gcc g++ python python-dev mercurial bzr gdb valgrind
gsl-bin libgsl0-dev flex bison tcpdump sqlite sqlite3 libsqlite3-dev libxml2
libxml2-dev libgtk2.0-0 libgtk2.0-dev uncrustify doxygen graphviz imagemagick
texlive texlive-latex-extra texlive-generic-extra texlive-generic-recommended
texinfo dia texlive texlive-latex-extra texlive-extra-utils texlive-generic-
recommended texi2html python-pygraphviz python-kiwi python-pygccxml
```

Next the latest NS3 software is downloaded from www.nsnam.com. The zip file was then saved to the Desktop for ease of access and then unzipped from the using the terminal command:

```
$ tar -xvf ns-allinone-3.30.1.tar.bz2
```

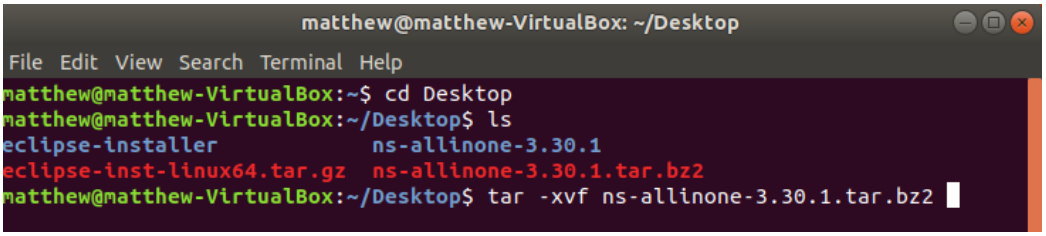

Figure 1: Desktop directory with ns3 unpacked

With the ns-allinone unpacked, what is included is a whole bunch of .cc files consisting partly of protocol examples. These must all be built and enabled. Once complete one can check for correct installation triggering test.py. Figure 2 displays summary of succesfull builds.

```
matthew@matthew-VirtualBox:~/Desktop/ns-allinone-3.30.1$ ./build.py
--enable-examples --enable-test

matthew@matthew-VirtualBox:~/Desktop/ns-allinone-3.30.1/ns-3.30.1$ ./test.py
```

```
Modules built:
antenna                  aodv                     applications
bridge                   buildings                config-store
core                     csma                     csma-layout
dsdv                     dsr                      energy
fd-net-device            flow-monitor             internet
internet-apps            lr-wpan                  lte
mesh                     mobility                 mpi
netanim (no Python)      network                  nix-vector-routing
olsr                     point-to-point           point-to-point-layout
propagation              sixlowpan                spectrum
stats                    tap-bridge               test (no Python)
topology-read            traffic-control          uan
virtual-net-device       wave                     wifi
wimax

Modules not built (see ns-3 tutorial for explanation):
brite                    click                    openflow
visualizer
```

Figure 2: Successful builds.

The .cc files can be found in the /ns-allinone-3.30.1/ns-3.3.1/examples folder, and useful files for getting acquainted with ns3 which will be demonstrated along with a visual interface, are first.cc and second.cc. These lie within the ../tutorial/ folder. To execute there files, they must be copied to the /ns-allinone/ns-3.3.1/scratch folder and then can can be executed using the following command. In Figure 3 the process of a point to point protocol is displayed.

```
$ ./waf -run scratch/first
```

```
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
```

Figure 3: Execution of first.cc

The simulation is a basic exchange of data between a client and a server. The template can be fully configured in the source code of the .cc file. For this the C/C++ text editor program Eclipse has been installed for convenience. It was possible to start a NS3 project in Eclipse where all the files of NS3 are included in one project platform to allow easy access and edits to of all .cc files in one place. Figure 4 shows a screenshot of the Eclipse platform in use.

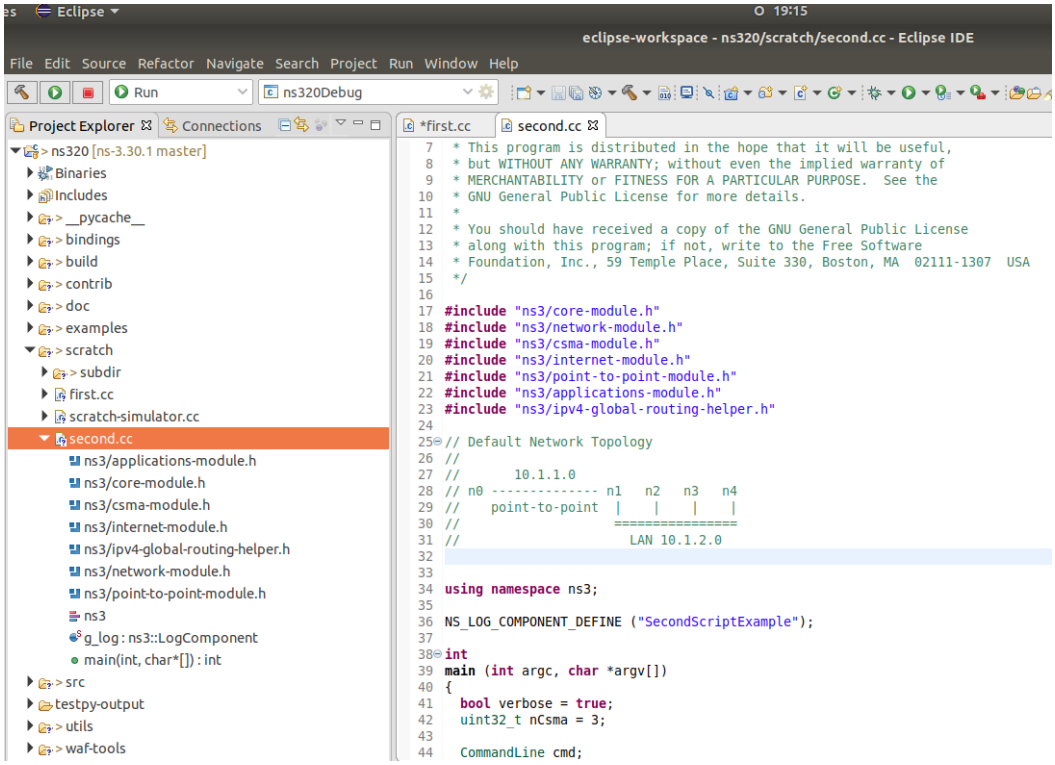

Figure 4: Eclipse platform

In this case the second.cc file is brought up. This program in an extension of first.cc. It is a default network topology consisting of a point-to-point between nodes n0 and n1 with a base address 10.1.1.0. and then a further connecting between n1 to CSMA nodes n2/n3/n4 using a LAN with base address 10.1.2.0.

Highlighting some of the remaining code, generation of CSMA node/devices, p2p node creation and starting and stopping the client connection is defined along with start and stop of the simulation are presented in figures respectively.

```
bool verbose = true;
uint32_t nCsma = 3;

CommandLine cmd;
cmd.AddValue ("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
```

Figure 5: Define number of CSMA devices on LAN

```
NodeContainer p2pNodes;
p2pNodes.Create (2);
```

Figure 6: Creates 2 p2p nodes

```
ApplicationContainer clientApps = echoClient.Install (p2pNodes.Get (0));
clientApps.Start (Seconds (2.0));
clientApps.Stop (Seconds (10.0));

Ipv4GlobalRoutingHelper::PopulateRoutingTables ();

pointToPoint.EnablePcapAll ("second");
csma.EnablePcap ("second", csmaDevices.Get (1), true);

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

Figure 7: Running simulation

3 NS3-GYM research paper analysis

The research paper written by Piotr Gawlowicz and Anatolij Zubow is in broad terms about how to use Machine Learning to optimise a network environment. The environment can be setup using the NS-3 simulator, which already has numerous environment examples to explore. Also these environments are flexible for the user to change and custom environments can also be setup. So the NS-3 platform is a great foundation for creating any real life network simulation. Reinforcement learning(RL) develops the algorithms that can then adapt and fit to its surrounding environment. To optimise algorithms they must all be tested and compared, and this is where the OpenAI Gym toolkit comes into action. The OpenAI can do exactly this and is supports numerous applications and programming languages.

The authors set upon the project with the goal to facilitate and shorten the time required for prototyping of novel RL-based networking solutions. The design allows for this as it can run multiple NS-3 instances, that can be adapted on the fly without creating a debugging issues. The outcome is an NS3-GYM toolkit which optimizes and speeds up the use of RL techniques for solving networking problems.

4 Conclusion

At this point it has been possible to successfully install and setup the NS-3 program on a ubuntu linux system hosted by Virtualbox on a Macbook Pro. There were several tutorials out there to help through the process, and the installation took a few attempts before it finally succeeded with optimal setup, including NetAnim, Synaptic and Eclipse programs which all assist in using the NS-3. Its early days but from running a few simple examples it has been possible to get an initial idea of how the NS-3 works. NS-3 seems to have many possibilities within network simulation, and the two gentleman who wrote the research paper seem to have found great purpose for the simulation tool aswell.

5 References

NS-3 Tutorials:
https://www.youtube.com/playlist?list=PLRAV69dS1uWQEbcHnKbLldvzrjdOcOIdY
www.nsnam.org