Introduction to Network Simulation

Getting started with NS3

Lab 5.1: CS3210 - Computer Networks Lab

Instructor: Krishna M. Sivalingam
Task I Due Date: 11.55PM, 8th March, 2015
Jan – May 2015, Indian Institute of Technology Madras

Introduction

In this assignment, the goal is to understand and experiment with network simulation using the NS3 software libraries. In the process, you will gain familiarity with setting up custom topologies in the simulation framework, setting up the protocol stack on the nodes, creating traffic sources, generating logs and plots. You will also learn how to write your own custom protocols and modules and how to integrate them into the simulation system. Overall, this will set the context for the second part of the assignment (Lab 5.2), which will be released the coming week. Some ground rules:

- This is an INDIVIDUAL assignment. You are allowed to refer to as many online resources as you need to get your program working – Please cite all the references that you used: especially those that helped you in finishing your code.
- It is strongly recommended that you have install a standalone version of Ubuntu/Linux in your computers if you do not already have it done (No Windows, Virtual Machines in Windows, etc.). This will also help with future assignments. If you use any OS other than a standalone Linux, the TAs may be unable to help you – you are on own with respect to debugging issues and getting the system working.

1 What is NS3?

In order to study or analyze newly developed protocols/algorithms/mechanisms, network simulators are commonly used. The simulator provides a controlled environment for development and testing. After success in the simulator, one would implement in a real system (e.g. make kernel level changes to TCP code) for further analysis.

NS3 (Network Simulator 3, available at `https://www.nsnam.org/`) is a popular discrete-event simulator. It is typically used for testing and evaluating networking protocols and strategies that are still in research phase. NS3 is essentially a set of libraries written in C++ (other than a few python bindings). Using the NS3 headers in your C++ programs, you can directly summon and construct the functionalities of a custom defined network. Usually, programs written using NS3 have two phases: (1) Defining the system - setting up the topology, and setting the various parameters that define the setup in each of the nodes (2) The actual simulation (including logging, packet capture, etc.).

You will obtain a better understanding of how it works once you see a few examples.

2 Downloading and Installing NS3

NS3 comes as a all-in-one package, which you can download and setup easily.

- Download the NS3 **All-in-one** package from here: `https://www.nsnam.org/ns-3-dev/download` (You will need mercurial if you want to clone the ns3 repository)

  [See more instructions at: `https://www.nsnam.org/wiki/Installation`]

- Run `./download.py` in the `ns3-allinone` folder to download the additional files.

- Configure NS3 with the command
  `./waf configure -enable-examples -enable-tests -disable-python`.

  Waf is a build system similar to ant or make. It is also used by NS3 as an interface to running simulations. The `-disable-python` part of this command disables the python bindings that allow simulations to be written in python. We will be using only C++ for the course; the NS3 python bindings are not guaranteed to work properly.

- Run `./waf` to actually build NS3. This process may take some time.

- You can now run the NS3 examples using `./waf --run <example_name>`. You can do a `./waf --run` to see the list of available modules to run. Of course, you must know what the example module does in advance so as to expect the corresponding outputs.

- If you wish to, you can configure NS3 to work with Eclipse using the following steps: `https://www.nsnam.org/wiki/HOWTO_configure_Eclipse_with_ns-3`

- More detailed documentation about the various classes is available at: `https://www.nsnam.org/docs/release/3.6/doxygen/modules.html`

3 Tutorial Tasks

- Follow the tutorial here to set up a UDP Echo client and server on a two-node topology.

- Follow the tutorial here to learn logging and packet-level capturing in the simulation.

- Generate the trace file and use wireshark/tshark/tcpdump to process the trace file.

- (Optional) Follow the tutorial here to learn more about tracing;

  Please carefully go through the above tutorials and get a feel for how things work in this framework.

4 Assignment Task I

In this task, you will create files in the `ns-3-dev/scratch` by copying files from the directory, `ns-3-dev/examples/tutorial/`. This directory has files called first.cc, second.cc, etc.; copy first.cc into `scratch` as newfirst.cc, etc.

- Let N denote the last three digits of your roll number. In **newfirst.cc**: (i) change the network address part to 10.1.N.0, e.g. student with roll number CS10B060 would use network 10.1.60.0; (ii) Change the DataRate to N Mbps; (iii) Change the Delay to N ms; (iv) change the maximum number of packets sent by the Echo client to N.

Next, enable Pcap on the point-to-point link by adding:

pointToPoint.EnablePcapAll("scratch/newfirst");

(
See also examples/tutorial/second.cc);

Run the simulation and obtain the pcap file. Verify correctness of messages using Ascii output to screen and reading the pcap file with wireshark or equivalent.

- Copy ns-3-dev/examples/tutorial/second.cc to **newsecond.cc**

  This code simulates a topology with 2 sets of nodes: one pair of nodes connected by a P2P link and the other set by a shared CSMA (such as Ethernet) network, with a common node between the two sets.

  Your task is: (i) In **newsecond.cc**, repeat the changes made to newfirst.cc; (ii) Change the total number of Csma nodes to 7, including n1; (iii) run another EchoServer on node 6 (n6) and Echoclient on node 2 (n2); (iv) Add the following lines, above the pcap enabling lines:

```
AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll (ascii.CreateFileStream ("scratch/newsecond.tr"));
```

  Run the simulation and obtain the traces and pcap file. Verify correctness of messages using Ascii output to screen and reading the pcap file with wireshark or equivalent.

Deliverables: Name your directory as **LAB5-CSnnBabc** and copy ONLY the newly created files (inside scratch/ directory) and pcap/output files, NOT the entire distribution. The tar-gzipped file should be uploaded to Moodle by March 8, 11.55PM.

The demo showing completion of Task I activities should be shown to the TA by Monday, March 9, 4 pm in the Lab.

5 Assignment Task II

Task II details will be provided before the Lab session starts on Monday, March 9. This will deal with understanding TCP congestion window behaviour and comparison of different TCP flavors.