CS169: Mobile Wireless Networks
Lab1: NS3 Introduction

Yue Cao and Azeem Aqil

advised by
Prof. Srikanth V. Krishnamurthy

University of California, Riverside
Jan 5 2014
1 General overview of the lab
   - Syllabus
   - Goals
   - What is NS3?

2 Session 1: NS3 Installation
   - Download and Build

3 Session 2: NS3 Configuration
   - First Simulation Scenario
1. We have 5 or 6 Lab and 1 Final Project. After the first 5 or 6 Lab, you don’t need to attend the Lab. I will assign a new office hour online for you to turn in homework and ask questions.

2. Turning in your homework at the beginning of the LAB. Solution will be posted after the LAB online.
The **general goals** of our NS3 Labs can be summarized in the following points:

1. Getting to know NS-3 which is a relatively new and powerful networks simulation tool.

2. Using NS-3 to simulate the protocol taught in Class and measuring the performance by changing the protocol. Have a deeper view with the protocol.
What is NS3?

1. NS-3 is an open source discrete event simulator used for networking research and education.

2. NS-3 is written entirely in C++.
   - User code, protocols and scenarios, also in C++.
   - Python wrappers for user code also exist.
Simulator Architecture

Looks just like IP architecture stack
Simulator Architecture

- Nodes may/may not have mobility.
- Nodes have network devices
  - Network devices transfer packets over channels.
  - Incorporating Layer 1 (Physical) and Layer 2 (Link).
- Devices interface w/ Layer 3 (Network: IP, ARP).
- Layer 3 supports Layer 4 (Transport: UDP, TCP).
- Layer 4 is used by Layer 5 (Application) objects.
Download and Build

1. Open a terminal and ssh to YourUserName@sledge.cs.ucr.edu.
2. Create a folder called ns3lab ($ mkdir ns3lab).
3. Go inside the folder ($ cd ns3lab).
4. Fetch the 3.13 NS-3 version (it’s enough for our following labs. You can check the new features with the newest version: http://www.nsnam.org/ns-3-21/)
Download and Build

1. Go to ns-3-allinone directory under ns3lab directory (command `cd`).

2. Make sure you have the following files inside your ns-3-allinone directory ($ ls): `-build.py*`, `-ns-3.13/`, `-pybindgen-0.15.0.795/`, `-util.py`, `-constants.py`, `-nsc-0.5.2/`. 

3. Now, let’s build NS3. Simply we will run the python build file. ($ ./build.py). Otherwise, we need to run configure, make and make install.
Glance at the first Script
In this exercise, we will learn how to configure NS3 and see two simplest scripts. The build system Waf is used on the ns-3 project. It is one of the new generation of Python-based build systems. You will not need to understand any Python to build the existing ns-3 system.

- Go inside ns-3.13 folder and type ($ ./waf configure).
- Try to type ($ ./waf). **Can you explain what is happening?**
- To validate you need to run ($ ./test.py) **You need to enable tests first. Use command ./waf --enable-tests configure.**
- Instead run this command ($ ./waf --run scratch/scratch-simulator)
Simple Simulation Scenario

Implement the local area network topology (exercise one). Create two tcp flows as the following,

- 10.0.0.3 → 10.0.0.5
- 10.0.0.2 → 10.0.0.4

Your simulation should last for 15 seconds. Hint (You should use examples/tutorial/first.cc as a reference code, which is explained at the following slides).
### Simple Script

- **You will need the following includes...**
  ```
  #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; The ns-3 project is implemented in a C++ namespace called ns3. This groups all ns-3-related declarations in a scope outside the global namespace, which we hope will help with integration with other code.**

- **You need to register some components (We will Talk about later in details)**
  ```
  LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
  ```

Yue Cao and Azeem Aqil  
CS169: Mobile Wireless Networks
**NS_LOG_COMPONENT_DEFINE("SimpleScript");** A statement that is need to generate documentation.

- **Create a Node Container (Create number of nodes)**
  ```cpp
  NodeContainer nodes;
  nodes.Create (2);
  ```

- **In order to create a point to point link, we need to create a point to point helper.**
  ```cpp
  PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
  ```

- **Create network devices**
  ```cpp
  NetDeviceContainer devices;
  devices = pointToPoint.Install (nodes);
  ```
Simple Script

- We now have nodes and devices configured, but we don’t have any protocol stacks installed on our nodes. The next two lines of code will take care of that.
  
  ```
  InternetStackHelper stack;
  stack.Install (nodes);
  ```

- **We assign Ip addresses**
  
  ```
  Ipv4AddressHelper address;
  address.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer interfaces = address.Assign (devices);
  ```

- **Create a server**
  
  ```
  UdpEchoServerHelper echoServer (9);
  ApplicationContainer serverApps = echoServer.Install (nodes.Get (1));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));
  ```
Simple Script

- **Create a client**
  UdpEchoClientHelper echoClient (interfaces.GetAddress (1), 9);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (1000));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (1024));
  ApplicationContainer clientApps = echoClient.Install
  (nodes.Get (0));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

- **Run simulation and Cal the destructor after finishing.**
  Simulator::Run ();
  Simulator::Destroy ();
The End.
The End.

Any questions?