CSCI 5280 – Software-Defined Networking

Lab 1

SDN Mininet-MiniEdit

University of Colorado Boulder

Department of Computer Science

Network Engineering

Professor Levi Perigo, Ph.D.

# Lab Summary

Mininet is the industry leading software network simulator used for SDN testing and troubleshooting. This lab will install Mininet (with a POX controller), and setup a basic learning switch application. This lab is intended to demonstrate basic functionality and understanding.

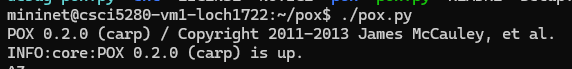
#NOTE: All commands that start with (#) or ($) must be entered in the Linux terminal. The commands that start with (mininet>) must be entered after starting Mininet from the Linux terminal. Review the following guides for additional information on Mininet and OpenFlow:

1. What is Mininet: <https://github.com/mininet/mininet/wiki/Introduction-to-Mininet>
2. Here is additional information on Mininet: <http://mininet.org/walkthrough/>
3. Here is additional information on OpenFlow and Mininet: <https://github.com/mininet/openflow-tutorial/wiki>

In this lab, you will also learn the basics of MiniEdit, which is the GUI of Mininet. MiniEdit can be used to configure Mininet topology, specify controller locations, and even run Mininet simulations. The intention of this lab is to expose the student to various options in SDN, specifically Mininet. Students are encouraged to use this lab as a guide only, and to expand on the topics for additional learning and experiments.

# Objective 1 – Connect to the Mininet VM and Install a POX SDN Controller

1. You will first be in the Linux shell of the VM.
2. Verify that the VM can ping your machine.
3. Verify that the VM has Internet connectivity by pinging 8.8.8.8.
   1. Note: If the previous steps do not work, you can't continue; you must have this working before continuing with this lab.
4. Issue the following command:
   1. $ sudo git clone <http://github.com/noxrepo/pox>
5. Provide a screenshot of the POX controller installed successfully. **[5 points]**



1. Instead of a POX controller, what are two other controllers you could have installed? **[5 points]**
   1. **I could have used Floodlight or Ryu**
2. If you had to choose any controller, which would it be and why? **[5 points]**
   1. **To my best knowledge, the POX controller is python based, which if I had to read some of the logic, I would be able to comprehend it better as compared to Floodlight which is primarily in Java.**

# Objective 2 - Run a Default Mininet Network

1. Initialize the default Mininet network topology. What command did you use? **[5 points]**
   1. I used ‘sudo mn’
2. Provide/"draw" a rough diagram of the default network that was created. **[5 points]**

A diagram of a structure

Description automatically generated

1. What are three network topologies that can be used (except sudo mn) by default in Mininet (provide the commands)? **[5 points]**
   1. **Sudo mn –topo linear**
   2. **Sudo mn –topo minimal**
   3. **Sudo mn –topo single**

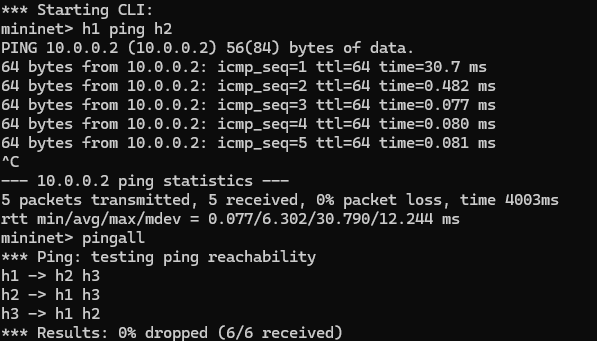
# Objective 3 – Implement a Programmable Layer 2 Switch and Remote Controller in Mininet

1. Open two terminal windows to the Mininet VM. (One will be used for Mininet and the other will be used for the POX controller)
2. On the first terminal, run the command:
3. # sudo mn --topo=single,3 --mac --controller=remote --switch=ovsk
4. Breakdown and explain what this command does in detail? **[5 points]**
   1. **–topo single,3: creates three single hosts**
   2. **–mac: automatically sets the host mac addresses**
   3. **–controller=remote: sets the type of controller, in this case the remote controller will not be generated from mininet, I assume we will use POX in future objectives for the controller.**
   4. **–switch=ovsk: sets the type of vswitch, in this case we have the ovsk switch.**
5. Try to ping the created nodes
   1. mininet> h1 ping h2
6. Will it ping? Why or why not? **[5 points]**
   1. **It does not ping because we set our controller to remote, which asks for an external controller, and we don’t have that set up yet.**
7. On the second terminal, run the POX controller now. Enter the following command:

$ sudo ~/pox/pox.py forwarding.l2\_learning \ openflow.spanning\_tree --hold-down \ log.level --DEBUG samples.pretty\_log \ openflow.discovery host\_tracker \ info.packet\_dump

1. Explain each component in the above POX command. **[10 points]**
   1. forwarding.l2\_learning: This component makes OpenFlow switches act as a type of L2 learning switch (1)
   2. openflow.spanning\_tree --hold-down: Turns on spanning tree for ports on the tree. Hold down makes sure all ports have a chance to be discovered. (1)
   3. log.level --DEBUG samples.pretty\_log: Setting the logging infrastructure to a pretty log that formats logs into a readable format (1)
   4. openflow.discovery host\_tracker: Sends LLDP messages out of OpenFlow Switches so that I can discover the network topology Host\_tracker keeps tracks of hosts in the network. If something changes, it will raise a HostEvent flag. (1)
   5. info.packet\_dump
   6. All descriptions are paraphrased from the documentation:
      1. (1) https://noxrepo.github.io/pox-doc/html/#
2. Now on the first terminal, try to ping the nodes again
   1. mininet> h1 ping h2
   2. mininet> pingall
3. Were the nodes able to talk to each other? Why? Why not? Provide a screenshot **[10 points]**

**They were because the remote controller POX can send ICMP packets between the hosts through given instructions to the switch.**



* 1. Check the debugs on the second terminal (with the POX controller). You should see the ARP entries that the switch has learned. Provide a screenshot of the debug information **[10 points]**

A screenshot of a computer program

Description automatically generated

# Objective 4 – Mininet Tasks and CLI Commands

# Part (a): Basic Mininet commands

1. Provide a screenshot of only the hostnames in the Mininet created above (include the command used). **[5 points]**
   1. **nodes**
2. Provide a screenshot of all the hosts devices in the Mininet, which includes their hostnames and IP addresses (include the command used). **[5 points]**
   1. Dump

A computer screen with white text

Description automatically generated

1. How would you see the route-table for the switch in the network? Provide a screenshot (include the command). **[5 points]**

A computer screen with white text

Description automatically generated

* 1. sudo ovs-ofctl dump-flows s1

## Part (b): Additional Mininet features

1. Initialize the default mininet topology. What is the average RTT when h1 pings h2? [**2 points**]
   1. ~0.1 ms
2. Now initiate a mininet topology using-

$ sudo mn --link tc,bw=10,delay=10ms

Explain this command. [**5 points**]

* + - This command is delaying the ping by 10ms at each link.

1. What is the average RTT when h1 pings h2 now? Explain the reason behind this. [**5 points**]
   1. Between each link we are delayed by 10ms. So h1 to s1 is 10ms, s1 to h2, h2 to s1, s1 to h1. Which adds up to ~40ms
2. Mininet allows us to use either the kernel-space switch or the user-space switch. Explain the difference between the two. [**5 points**]
   1. The kernel space switch runs within a VM and is faster when it comes to the ping tests. The user-space switch is controlled by the OS itself and processes has to be scheduled through the OS which is slower.
3. Execute these commands one after the other and provide the snapshots of their results-

sudo mn --switch ovsk --test iperf

sudo mn --switch user --test iperf

A screenshot of a computer program

Description automatically generated

Also, explain the results. [**10 points**]

* + - As we can see, the throughput of kernel-space is much higher than user-space by a huge margin. It shows that running through the kernel is more efficient than through the user OS.

# Objective 5 – Update Mininet to Latest Version

1. List the command you would use to check the current version of Mininet in the VM **[5 points]**
   1. **Mn --version**
2. PuTTY/SSH in to the Mininet VM and execute the following commands:

$ cd mininet

$ git remote -v

$ git remote set-url origin <https://github.com/mininet/mininet>

$ git fetch

$ git checkout master

$ git pull

$ sudo make install

1. What is the git command used for? **[5 points]**
   1. **The git commands were to pull the latest version of mininet from github.**
2. Provide a screenshot that shows that the installation script is successful as well as the current version of Mininet **[5 Points]**

A screenshot of a computer program

Description automatically generated

Objective 6 – X11 Forwarding: Setup the SDN (controller/switch) Environment

* Download Xming (Windows) or Xquartz (MAC).
* Run Xming software on local machine.
* Start a new PuTTy/SSH session to the Mininet VM.
* Before you connect, in PuTTy go to Connection > SSH > X11
* Check the “Enable X11 forwarding” box.

[Note: If using MAC/Linux you can simply SSH with the “-X”.]

* Initiate the PuTTy connection.
* Verify that your local machine can talk to the Mininet VM.

# Objective 7 – Use MiniEdit to Build a Simulation Topology

1. PuTTy/SSH in to Mininet VM with X11 forwarding enabled.
2. Execute the following command:

sudo ~/mininet/examples/miniedit.py

1. Provide a screen shot of MiniEdit running in X11 **[5 points]**

A white board with a black border

Description automatically generated

1. Use the GUI to build a topology as is shown in the figure 1. below. All links have bandwidth of 20Mbps.

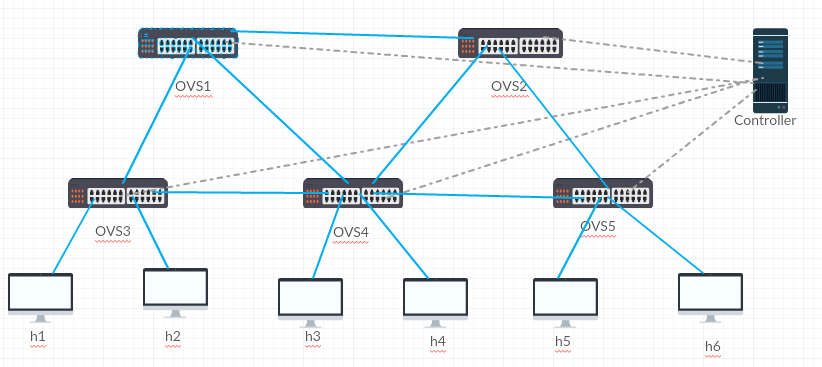


Figure 1. Mininet topology for Lab1

1. Start POX controller in a separate terminal of Mininet VM and clear any of the previously run Mininet topology. Provide screenshot of the controller running **[5 points]**

A computer screen with white text

Description automatically generated

1. Set all the links to 20Mbps and point the controller to the correct IP and port number
   1. Provide a screenshot of your topology inside MiniEdit **[5 points]**

A screenshot of a computer

Description automatically generated

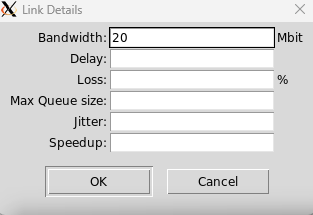
1. Provide screenshots of the different property windows of Controller, switch, host, and links **[20 points]**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

1. In the Preference of MiniEdit select “Start CLI” option
2. Save the topology as .py topology file using “Export Level2 script,” attach the .py file with your report **[10 points]**

# Objective 8 – Run Simulation with MiniEdit

1. In MiniEdit click “RUN”
2. Provide a screenshot of the results **[10 points]**

A screenshot of a computer program

Description automatically generated

1. Run “pingall” in the CLI
2. Open a root terminal from MiniEdit or the switch CLI
3. Execute the command which shows all flows for switch 1
   1. Provide a screenshot of the command and output **[10 points]**

A black text on a white background

Description automatically generated

sudo ovs-ofctl dump-flows s1

# Objective 9 – Simulate a Broken Link

1. In the MiniEdit GUI, mark the link between “OVS3 and OVS1” as well as “OVS3 and OVS4” as “down” using the right click menu.
   1. Provide a screenshot of the updated topology **[10 points]**

A computer screen shot of a network

Description automatically generated

1. Run “pingall” in the Mininet CLI, provide a screenshot of the results **[5 points]**

A screenshot of a computer

Description automatically generated

Objective 10 – Report Questions

1. Explain five advantages of using Mininet **[5 points]**
2. What are two limitations of Mininet **[2 points]**
3. What exactly is POX and what do you think are its advantages? Do you think POX is actively used in the industry? Why/why not? [**5 points**]
4. Like MiniEdit, what other tool can be used for creating customized Mininet topology scripts? **[3 points]**

# Total Score = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/222

# 