ARIZONA STATE UNIVERSITY

CSE 434, SLN 70516 — **Computer Networks** — Fall 2022

Lab #4

Due electronically before 11:59pm, Sunday, 12/04/2022

This lab consists of an introduction to OpenFlow using POX (python based) controllers, on CloudLab. To ease grading:

1. Include your group number and group member(s) on the title page of your report.
2. Label each exercise by its number, and provide your solution to the exercises in numerical order.
3. Your report must consist of a **single file**, *i.e.*, you must import any figures and/or screen shots directly into your report. Remember to always include your ASUrite id in any screen shot you take.

1 An Introduction to OpenFlow with POX Controllers

In the introduction to OpenFlow tutorial, after configuring the switch and seeing the operation of a learning switch, you will first experiment with three simple POX controllers:

1. A *traffic duplicator controller* duplicates all the traffic of the OpenFlow switch out a specific port.
2. A *TCP port forwarding controller* diverts all traffic destined to host A on TCP port X, to the same host A on TCP port Y.
3. A *proxy controller* diverts all traffic destined to host A on TCP port X, to host B on TCP port Y.

You are provided with an implementation for each of these controller programs. However, if you are so inclined, you can try implementing these controller programs yourself using the utilities provided; see step **3d. Download the pox apps** in **Step 3. Execute Experiment**.

You will be required to implement one controller program of your own to:

4. A very simple *denial-of-sevice controller* that drops all traffic from a specific source IP address.

Figure 1 shows the topology you will set up. In general, the controller needs to have a public IP address, so that it can exchange messages with the OpenFlow switch; in this tutorial, we use an OpenFlow Software Switch, Open vSwitch (OVS). The controller for the switch can run anywhere in the Internet; we use POX, a python-based controller framework, which is just one example of many controller frameworks.

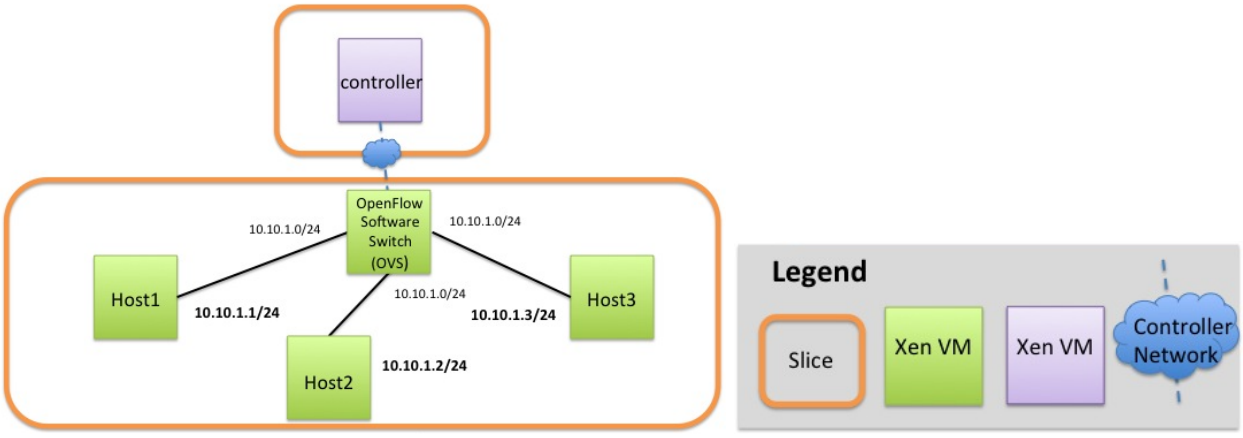

Figure 1: The CloudLab topology used: It has two slices and uses Xen VMs.

host file -> overrides what the DNS server will return by providing your own DNS mapping

1.1 Experiment Set-up

The tutorial Introduction to OpenFlow Tutorial (OVS) with POX Controller has been ported from GENI (another testbed) to CloudLab.

Almost all of the instructions are the same, except:

1. In **Step 1. Obtain Resources** of **Part I: Design Setup**, rather than allocate resources on GENI, you will instantiate the topology on CloudLab. To start the experiment, click on to instantiate the profile on CloudLab, *i.e.*, to obtain the resources corresponding to the topology in Figure 1. For interest, choose one cluster for the VM that runs the OpenFlow controller, and a different cluster for the network that includes a VM with OVS installed.

   The IP addresses assigned may be different from those shown in Figure 1. If so, you may need to adapt the instructions that follow accordingly.

2. In **Step 2. Configure and Initialize**, after logging in to the OVS host (Step 2.a.i), issue the following command on the OVS host:
   `sudo apt install openvswitch-switch`

In step- 2a, we are using a bridge to connect between the host and OVS which helps to be able to forward and connec to the OVS's eth1 - eth3

You should be able to run the rest of the tutorial, *i.e.*, from Step 1.a.ii onwards, by following the instructions without modification.

sudo ifconfig ethX 0 -> the 0 means to remove the inet and replace with 0.0.0.0

Exercise 1.1: For your lab report: In Step 2 (Configure the Open vSwitch), Step 2a (Configure the Software Switch (OVS Window)), part v, take a screen shot for your report that shows that your software switch is configured. Explain by referencing your screen shot.

Exercise 1.2: In Step 2, Step 2b (Point your Switch to a Controller), part v, take a screen shot for your lab report that shows that your switch is pointed to a controller. Explain by referencing your screen shot.

Step 2b sets the `fail-safe-mode` to `secure`, as described in Step 2c. Follow this by Step 3 (Execute Experiment), Step 3a, logging in to your hosts.

Exercise 1.3: Use a Learning Switch Controller.
For your lab report:

1. In Step 3b (Use a Learning Switch Controller) step 3, take a screen shot that illustrates the operation of the learning switch controller, *i.e.*, shows the `ping` output between `host1` and `host2`.
2. In Step 3b step 4, take a screen shot of the flows installed on the switch by the controller.

For both screen shots, provide a brief explanation of what you observe in your screen shot.

Ether is the MAC address

Steps 3c and 3d are optional: You may be interested to follow the instructions in Step 3c, to look around your OVS switch. If you interested to try develop your own controller program, you may want to read Step 3d, on debugging a controller.

Exercise 1.4: Run a Traffic Duplication Controller.

1. In Step 3e, after executing the `curl` command, make a note of the details of your topology such as the Data Path IDentifier (DPID) of the Open vSwitch, the MAC addresses of the hosts, and the port numbers on which the hosts are connected to the Open vSwitch, *etc.* These are highlighted in yellow in the instructions.
2. Use the information collected in Step 3e in Step 3f (Run a Traffic Duplication Controller), to insert a flow to duplicate traffic. Take a screen shot for your lab report that illustrates the operation of the traffic duplication controller, *i.e.*, the `tcpdump` output that shows duplication is happening. Provide a brief explanation of what is observed in your screen shot.

Exercise 1.5: Run a Port Forwarding Controller.

1. Use the information collected in Step 3e in Step 3g (Run a Port Forward Controller).
2. Take a screen shots for your lab report in step 3 showing the `netcat` output before rule for the port forwarding is inserted.
3. Take another screen shot for your lab report in step 6, that illustrates the operation of the port forwarding controller, *i.e.*, showing the `netcat` output after the port forwarding is inserted.

Provide a brief explanation of what is observed in each screen shot.

Exercise 1.6: Run a Server Proxy Controller.

1. Use the information collected in Step 3e in Step 3h (Run a Server Proxy Controller).
2. Take a screen shots for your lab report that illustrate the operation of the proxy controller, *i.e.*, before the traffic is diverted, and in step 6, showing the diverted traffic.

Provide a brief explanation of what is observed in the screen shots.

Review the programs that implement the *traffic duplicator controller*, *TCP port forwarding controller*, and *proxy controller*. Use them as models to implement and test your own *denial-of-service controller*.

Exercise 1.7: Implement your own Denial-of-Service Controller.

1. Select one of the hosts to block, say Host2.
2. Implement a *denial-of-service controller* that drops all traffic coming from the source IP address of the host you selected. (The source IP address can be hard coded.) You should make use of the utilities provided; see step **3d. Download the pox apps** in **Step 3. Execute Experiment**.
3. Upload and test your controller program. It should allow traffic from Host1 and Host3 and block the traffic from Host 2.
4. Design an experiment to demonstrate the functionality of your controller, taking a screen shots for your lab report that illustrate its operation, *i.e.*, both blocked and unblocked traffic.

Include the code for your controller program, as well as an explanation of the experiment you designed to demonstrate its correctness. Also provide a brief explanation of what is observed in the screen shots that are part of your demonstration.

Before tearing down your experiment (Step 4), delete your bridge (see Step 3i).

Replace the proxy.py with the denialofservice.py we have written