BLG632 - Next Generation Wireless Networks

**Homework Assignment #6- Part 1**

**Due Date: May 19 Friday 17:00**

**TOTAL POINTS: 10**

This homework is for you to LEARN the following:

- How to create a simple custom topology in mininet

- How to run a POX server on the same VM that provide rules to your switch in mininet

- How to add different flow entries into a switch using POX controller

- How the rules can match on different layers, etherenet, IP, TCP

Please try to do this homework on your own. If you are stuck you can ask your colleagues or Muge (but wait and try on your own before asking for direct help).

DO NOT COPY CODE FROM EACH OTHER. **DO NOT.**

The homework is based on the following two videos of David Mahler.

Video1: Creating custom topologies: <https://www.youtube.com/watch?v=yHUNeyaQKWY&t=361s>

Video2: Openflow entries on Open vSwitch: <https://www.youtube.com/watch?v=FyV4MoQ3T0I&t=183s>

Most part of the homework is based on creating openflow entries in OVS using POX controller. In the video David uses “sh ovs-ofctl” commands to add entries to OVS in the mininet environment itself. Here, we will have a POX controller provide the rules instead to the switch. This shows you the power of Centralized controller by a simple one switch example.

Also, the two example files created and provided by Muge:

- example1\_topology.py and

- example1\_controller.py

will be good source for you in creating some of the topology and POX controller rules. You can copy paste from these two files.

The following file available on directory /home/vagrant/pox/pox/forwarding/hub.py will also be useful. This will form the starting point for your pox controller files.

You will need to work on the VM provided for the course. I suggest having four terminals: (a) for starting, stopping and cleaning up mininet, (b) for the pox controller, (c) for editing and updating the pox controller files (this could be a local terminal that ssh into the VM e.g. vagrant ssh. This may allow you to edit files on the VM easier), and (d) for wireshark (command: sudo wireshark &. (NOTE: in my case I could not have wireshark window show up remotely on my desktop. I got a display error when I typed “ssh –X [vagrant@192.168.0.100](mailto:vagrant@192.168.0.100)”; sudo wireshark &. Rather than trying to solve that I decided to have the wireshark display natively in the VM (terminal in the VM) rather than ssh into the VM).

NOTE: Whenever you exit out of mn or kill a controller, make sure you run “sudo mn –c” to clean up your VM environment.

Watch Video2 from 00:00 to 02:15

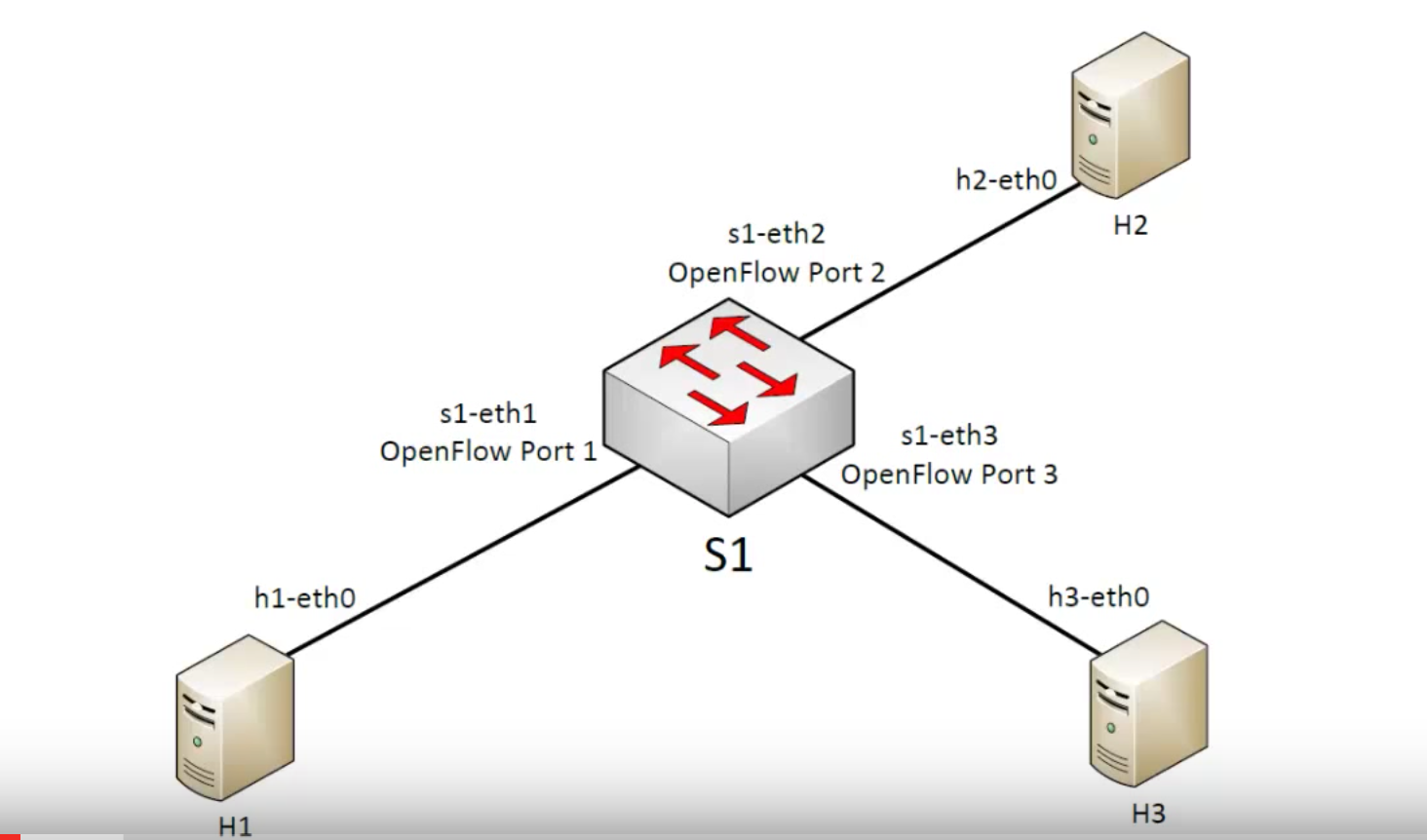
1. Create a simple 1 switch three host topology shown in Video2 and given below by creating a python program. Lets call this program s1h3topo.py. (hint: example1\_topology.py) Create this file in the folder /home/vagrant/mininet/example folder. Make sure that the IP and mac addresses are as follows:

H1 10.0.0.1, mac 00:00:00:00:01

H2 10.0.0.2, mac 00:00:00:00:02

H2 10.0.0.3, mac 00:00:00:00:03

You do not need to setup any remote controller, but use the default controller c0. Your program should bring you to your CLI in mininet, i.e. after you type sudo python s1h3topo.py you should find yourself in mininet. From there you should be able to use dump, net, nodes command to see your network. Also, use the sh ovs-ofctl dump-flows s1 to see the rules in switch s1. Check the difference in sh ovs-ofctl dump-flows s1 command before and after ping. How many rules do you see after pingall –c2



Exit out of mininet. Cleanup your environment: sudo mn -c

2. Now copy your s1h3topo.py to a file called s1h3topoRemote.py. This should be connecting to remote controller on your same VM, IP address 127.0.0.1 port=6633. (hint: see example1\_topology.py). We will use POX controller with the simplest forwarding rules, called hub.py. take a good look at the code in directory: /home/vagrant/pox/pox/forwarding/hub.py. This is a really simple file and basically says that when switch attaches to the controller send a rule: if any packet comes on any interface, send it to all other interfaces (flood). Best for you to copy this file hub.py into your local machine (or wherever you are going to edit your controller files). This file will become the basis for all your controller programs. In another terminal on your VM, first start the pox controller with the forwarding rule, i.e go to directory pox. Then type: /home/vagrant/pox:./pox.py log.level --DEBUG forwarding.hub. This will start your pox controller on loopback address (127.0.0.1) and port 6633. For more understanding of how this code works see: <https://github.com/PrincetonUniversity/Coursera-SDN/tree/master/assignments/simple-controller> . In another terminal /home/vagrant/mininet/examples:sudo python s1h3topoRemote.py. What do you see in the rules of your switch S1, i.e what is the output of sh ovs-ofctl dump-flows s1? Also in your wireshark you should be able to see that when the switch comes up the POX controller sends a single rule to the switch.

Kill your pox controller (CNTRL-C). Exit out of mininet. Cleanup your environment: sudo mn –c

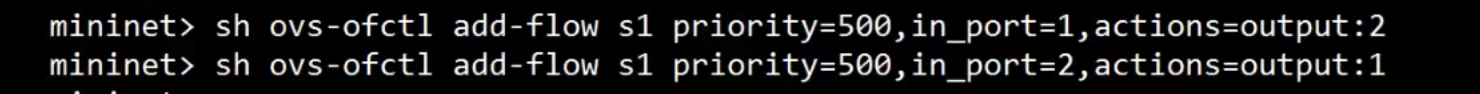
Watch Video2 from 02:13 to 03:45

3. Now we will create your first ever pox controller file ☺. Now we create the equivalent rule “sh ovs-ofctl add-flow s1action=normal”, but not inside mininet or from shell, but from your POX controller and send it to the switch. In one of the terminals in the VM cd pox/pox/misc. Copy the file hub.py to there: cp ../../forwarding/hub.py s1h3controllerNormal.py. Now you will edit the file such that the rule that is having action NORMAL and not a flood rule. You just need to change one field in the file s1h3controllerNormal.py (hint OFPP\_NORMAL). You can delete all the initial copywrite stuff at the start of the s1h3controllerNormal.py file. This lets you see the simple code better. Now in another window: cd pox, so that you are in folder /home/vagrant/pox. Start your controller: /home/vagrant/pox:./pox.py log.level –DEBUG misc.s1h3controllerNormal. In another terminal /home/vagrant/mininet/examples:sudo python s1h3topoRemote.py. What do you see in the rules of your switch S1, i.e what is the output of sh ovs-ofctl dump-flows s1? Also in your wireshark you should be able to see that when the switch comes up the POX controller sends a single rule to the switch. (My wireshark has a bug and does not show me the rule correctly. However sh ovs-ofctl dump-flows s1” shows me that the right rule is installed).

Kill your pox controller (CNTRL-C). Exit out of mininet. Cleanup your environment: sudo mn –c

Watch Video2 from 03:45 to 06:36

4. Port based rules: Now we will create the following two port based rules and send them from your POX controller to your switch S1. The two rules are shown below:

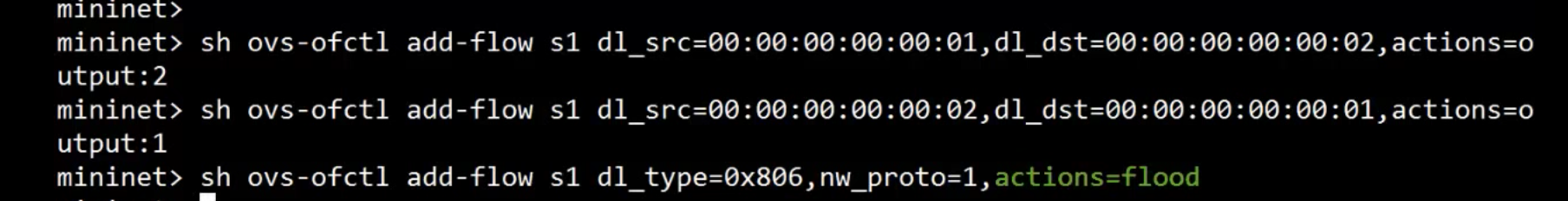


In pox/pox/misc folder: cp s1h3controllerNormal.py s1h3controllerPortForwarding.py. Edit the file to create the two rules to be sent to your switch (hint: look at example1\_controller.py to see how to create these port forwarding rules). When done from directory /home/vagrant/pox:./pox.py log.level –DEBUG misc.s1h3controllerPortForwarding. In another terminal /home/vagrant/mininet/examples: sudo python s1h3topoRemote.py.What do you see in the rules of your switch S1, i.e what is the output of sh ovs-ofctl dump-flows s1? Verify that you have the same behaviour as in the Video2 that you just watched.

Kill your pox controller (CNTRL-C). Exit out of mininet. Cleanup your environment: sudo mn –c

Watch Video2 from 06:36 to 08:52

5. Layer 2 matching rules: Now we will create the following three Ethernet addresses based rules and send them from your POX controller to your switch S1. The two rules are shown below:

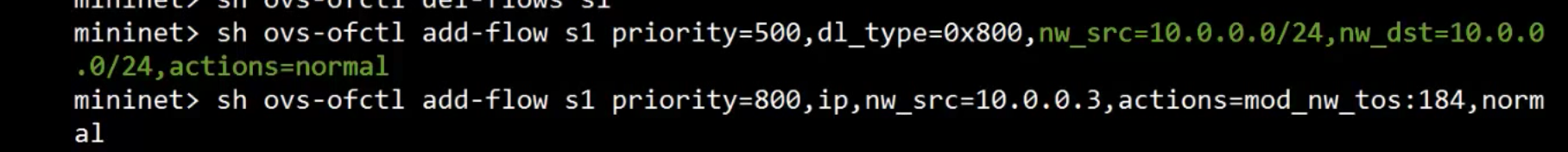


In pox/pox/misc folder: cp s1h3controllerPortForwarding.py s1h3controllerL2Forwarding. Edit the file to create the three rules to be sent to your switch [hint: from pox.lib.addresses import EthAddr and msg.match.dl\_src=EthAddr('00:00:00:00:00:01') ]. When done from directory /home/vagrant/pox, run ./pox.py log.level –DEBUG misc.s1h3controllerL2Forwarding. In another terminal /home/vagrant/mininet/examples: sudo python s1h3topoRemote.py.What do you see in the rules of your switch S1, i.e what is the output of sh ovs-ofctl dump-flows s1? Verify that you have the same behaviour as in the Video2 that you just watched.

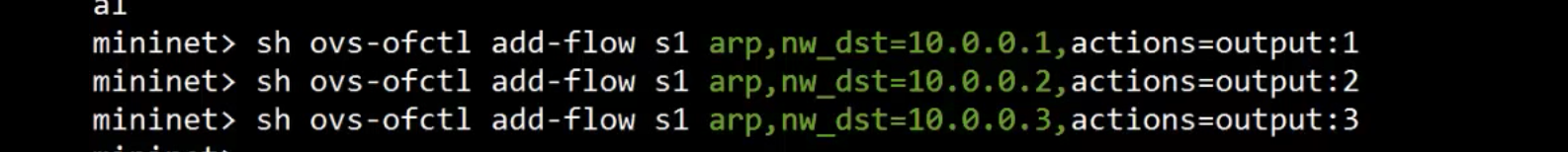
Kill your pox controller (CNTRL-C). Exit out of mininet. Cleanup your environment: sudo mn –c

Watch Video2 from 08:52 to 12:36

6. L3 matching rules: Now we will create the following two L3 addresses based rules and send them from your POX controller to your switch S1. The two rules are shown below:



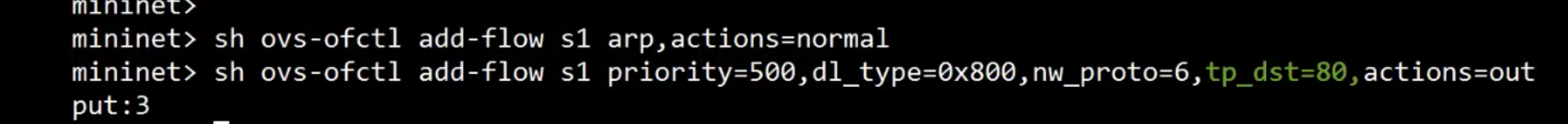
and the following three L2 matching rules:



In pox/pox/misc folder: cp s1h3controllerL2Forwarding s1h3controllerL3Forwarding. Edit the file to create the five rules to be sent to your switch [hint: look at example1\_controller.py. For modifying TOS, you will need to do some looking at internet and see how that can be achieved. If you cant modify TOS, do not get stuck, but move on after having given this a good try. You will get points for figuring out TOS modification]. When done from directory /home/vagrant/pox, run ./pox.py log.level –DEBUG misc.s1h3controllerL3Forwarding. In another terminal /home/vagrant/mininet/examples: sudo python s1h3topoRemote.py.What do you see in the rules of your switch S1, i.e what is the output of sh ovs-ofctl dump-flows s1? Verify that you have the same behaviour as in the Video2 that you just watched.

Watch Video2 from 12:36 to 14:22

7. L4 matching rules: Now we will create the following L4 based rules and send them from your POX controller to your switch S1. The two rules are shown below



In pox/pox/misc folder: cp s1h3controllerL3Forwarding s1h3controllerL4Forwarding. Edit the file to create the five rules to be sent to your switch [hint: look at example1\_controller.py]. When done from directory /home/vagrant/pox:./pox.py log.level –DEBUG misc.s1h3controllerL4Forwarding. In another terminal /home/vagrant/mininet/examples: sudo python s1h3topoRemote.py.What do you see in the rules of your switch S1, i.e what is the output of sh ovs-ofctl dump-flows s1? Verify that you have the same behaviour as in the Video2 that you just watched.

Please submit all the python programs: topology and POX controller for each of the above.

Hope this helped you understand and learn about POX controller and building and downloading rules to switches. This is not a controller that will be deployed in a general environment, but learning is always easiest and best when things are simple.