CSEN 241 HW 3
Mininet & OpenFlow

Learning Objectives

The aims of the assignment include the following learning objectives:

- Understanding how to use Mininet
- Understand how the control plane works & OpenFlow protocol via POX
- Understanding how to form a data center topology

Prerequisite

You are free to use your personal computer for this assignment. The set of requirements for using your personal computers is as follows:

- CPU with at least 2 cores
- 4 GB memory
- Docker
- OS: **Linux (Ubuntu preferred) as a VM (with x86 emulation if you are on M1 Mac)**

Step 1: Install and Run Mininet

What is Mininet?

Mininet [1] is a great way to develop, share, and experiment with OpenFlow and Software-Defined Networking systems. Mininet allows you to easily and quickly create various network topologies on your local machine.

Installing Mininet

For this exercise, you are expected to run all the mininet commands using a docker container. We can pull the Mininet docker container and start it simply by running the following commands:

```
$ sudo docker pull iwaseyusuke/mininet
$ sudo docker run -it --name=mininet --privileged iwaseyusuke/mininet
```

If you are using a machine with no Open vSwitch installed, run the following command in your machine, not in the container, to install Open vSwitch

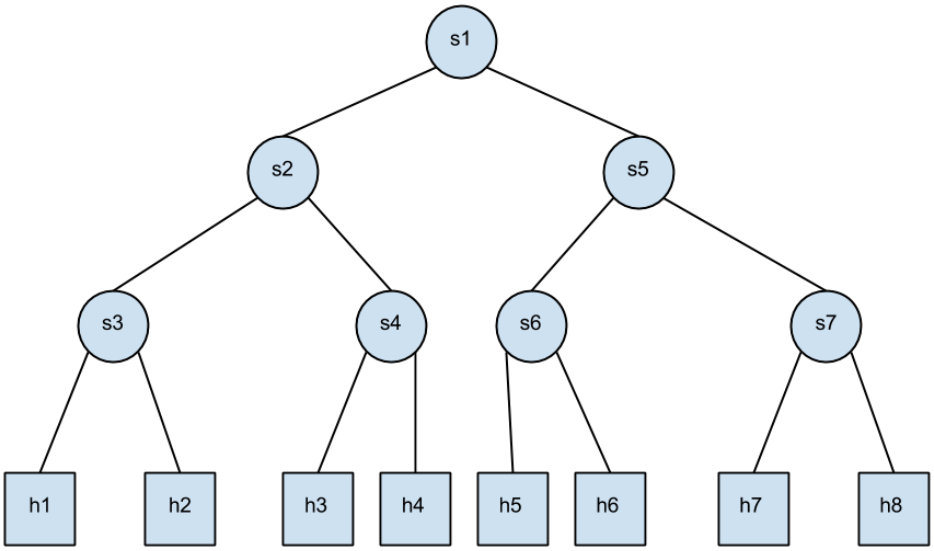
```
$ sudo apt update
$ sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder
bridge-utils
$ sudo apt install openvswitch-switch
```

Task 1: Defining custom topologies

As mentioned in the lecture, data center networks traditionally operate a Fat-tree topology. In a Fat-tree, End-hosts connect to top-of-rack switches, which form the leaves of the tree; one or more core switches form the root; and one or more layers of aggregation switches form the middle of the tree. An example of the fat-tree topology is shown in the figure below.

Task 1 is for you to develop a Python script that can mimic a fat-tree topology in Mininet. While it would be best to simulate a fat-tree topology, it would be too difficult to get it working for this assignment, so we will create a similar binary tree topology. The binary tree we are creating should have **7 switches and 8 hosts**, connected in the topology shown in the figure below. The starter code has been provided for you, called “binary_tree.py” and you can find how to write custom topologies in [2]. After completing the Python script, you should run the following commands to verify that h1 can ping h8.

```
$ sudo mn --custom binary_tree.py --topo binary_tree
mininet> h1 ping h8
```



Once you are sure that you have created the right topology, answer the following question.

Questions

1. What is the output of “nodes” and “net”
2. What is the output of “h7 ifconfig”

Step 2: Install POX

What is POX

POX [3] is an open source development platform for Python-based SDN control plane. For this assignment, we will use POX as the OpenFlow controller.

Installing POX

We will install POX in the Mininet Docker container as follows. In a new terminal run,

```
$ sudo docker exec -it mininet /bin/bash
```

Then download POX by running the following commands:

```
$ apt-get update
$ apt-get install git
$ apt-get install python3
$ env GIT_SSL_NO_VERIFY=true git clone https://github.com/noxrepo/pox
```

Once POX is installed via git clone, you can run POX by running the following commands:

```
$ ./pox.py log.level --DEBUG misc.of_tutorial
```

This loads the controller in *~/pox/pox/misc/of_tutorial.py*. **This controller will act like a “dumb” switch and floods all packets.**

Finally, you can connect Mininet to POX by running Mininet with the --controller remote arguments as follows:

```
$ sudo mn --custom binary_tree.py --controller remote --topo
binary_tree
```

You must run this while pox.py is running in a separate shell. Once you do this, this will connect all switches to the controller. Please take a look at the output from the controller console.

Task 2: Analyze the “of_tutorial’ controller

We have learned in lecture that, in SDN, if there are no forwarding rules for a particular packet installed in the switch, such a packet will be forwarded to the SDN controller. Review this concept by studying the “of_tutorial” controller, and answer the following questions.

Questions

1. Draw the function call graph of this controller. For example, once a packet comes to the controller, which function is the first to be called, which one is the second, and so forth?
2. Have h1 ping h2, and h1 ping h8 for 100 times (e.g., h1 ping -c100 p2).
    a. How long does it take (on average) to ping for each case?
    b. What is the minimum and maximum ping you have observed?
    c. What is the difference, and why?
3. Run “iperf h1 h2” and “iperf h1 h8”
    a. What is “iperf” used for?
    b. What is the throughput for each case?
    c. What is the difference, and explain the reasons for the difference.
4. Which of the switches observe traffic? Please describe your way for observing such traffic on switches (e.g., adding some functions in the “of_tutorial” controller).

Task 3: MAC Learning Controller

In this section, we will make the switches smarter, as the current switches can’t really do much but flood the packets they receive. Switches will learn the ports packets arrive from, and upon receiving a packet, if they have already seen its destination address, they will know the exact port to forward it on and avoid flooding the network (i.e., MAC learning).
Please add the following code into the “of_tutorial” controller, which enables MAC learning. Please change the source code in of_tutorial.py accordingly to make sure “act_like_switch” is active in your code (instead of act_like_hub), analyze the behaviors, and answer the questions.

```
def act_like_switch (self, packet, packet_in):
  # Learn the port for the source MAC
  # print("Src: ",str(packet.src),":", packet_in.in_port,"Dst:", str(packet.dst))
  if packet.src not in self.mac_to_port:
    print("Learning that " + str(packet.src) + " is attached at port " +
str(packet_in.in_port))
    self.mac_to_port[packet.src] = packet_in.in_port

  # if the port associated with the destination MAC of the packet is known:
  if packet.dst in self.mac_to_port:
    # Send packet out the associated port
    print(str(packet.dst) + " destination known. only send message to it")
    self.resend_packet(packet_in, self.mac_to_port[packet.dst])
  else:
    # Flood the packet out everything but the input port
    # This part looks familiar, right?
    print(str(packet.dst) + " not known, resend to everybody")
    self.resend_packet(packet_in, of.OFPP_ALL)
```

Questions

1. Describe how the above code works, such as how the "MAC to Port" map is established. You could use a ‘ping’ example to describe the establishment process (e.g., h1 ping h2).
2. **(Comment out all prints before doing this experiment)** Have h1 ping h2, and h1 ping h8 for 100 times (e.g., h1 ping -c100 p2).
    a. How long did it take (on average) to ping for each case?
    b. What is the minimum and maximum ping you have observed?
    c. Any difference from Task 2 and why do you think there is a change if there is?
3. Q.3 Run “iperf h1 h2” and “iperf h1 h8”.
    a. What is the throughput for each case?
    b. What is the difference from Task 2 and why do you think there is a change if there is?

Task Overview and Rubric

Here is the list of tasks to do. **All of the code and questions must be updated via github. Please reuse the existing repository.**
Files to upload

- binary_tree.py
- of_tutorial.py
- README.txt
    - README.txt should contain all the answers to your questions.

Grading Breakdown

- Task 1
    - binary_tree.py (30 pts)
    - Questions (10 pts, 5 pts each)
- Task 2: Questions (20 pts, 5 pts each)
- Task 3
    - of_tutorial.py (25 pts)
    - Questions (5 pts each)

Submission Instruction

Create a submission for the assignment on the Camino containing the following information:

1. a URL such that the command ‘git clone URL’ would download your repository (assuming that the repository has been shared with the requesting user);
2. a git commit ID within your repository that you want markers to assess that contains the code and the README.txt

You can reuse the existing repository to not have to re-invite the teaching staff

References

[1] An Instant Virtual Network on your Laptop (or other PC). http://mininet.org/
[2] Mininet Custom topologies. http://mininet.org/walkthrough/#custom-topologies
[3] POX. https://github.com/noxrepo/pox