SINGAPORE

ICTIOIO Computer Networks (AY2021/22) TECHNOLOGYINSTITUTE OF

Individual Assignment: Self-learning lab worksheet

Mininet Emulation: Software Defined Networks (SDN) — Control & Data Plane

LEARNING OUTCOMES

Upon completion, you should be able to:

- Understand the concept of software defined networking (SDN)
- Apply and infer the working of the SDN concepts, i.e., separation of data and control plane using Mininet Emulation Software with Floodlight controller

NAME: Gu Boyuan

STUDENT ID: 2103210

REQUIRED SOFTWARE

- VMware workstation (trial version) or VMware player or VirtualBox • Mininet
- Floodlight controller
- Wireshark/Tshark

Submission:

1. Fill this lab worksheet (pg 2 - 5), answer all questions within the box provided and upload to Gradescope by 20th Mar, 2022 (Sun) by 11.59 pm.
2. This is an individual lab exercise comprising of 20 marks and the weightage of this submission towards the continuous assessment of the module is 5%.
3. Clearly fill in your name, ID above in the space provided (in blue) before uploading this worksheet to the Gradescope. Otherwise, your submission will not be recognized by Gradescope. You may access Gradescope from the module LMS page or directly from the below link. https://www.gradescope.com/courses/376505

Partl — Setup Mininet with Default Controller:

Mininet is a network emulation tool widely used for educational purposes. The objective of Part 1 is to get a good understanding of Software Defined Networking, OVS, and the OpenFlow protocol through the Mininet emulator. Note that the OVS code running on the emulator is the exact replica of the code that is running on any OpenFlow supported hardware.

ICTIOIO Computer Networks (AY2021/22)

1.1. Download the Mininet VM 2.3.0 Ubuntu VM Image (latest release) https://github.com/mininet/mininet/releases/
1.2. Boot the Mininet VM and setup http://mininet.org/vm-setup-notes/
1.3. Log in to VM with username: mininet and password: mininet
1.4. Find the IP address of your VM using ifconfig
1.5. Come out of your VM (do not close the VM). From your host machine SSH to your mininet VM (e.g., ssh mininet@lPaddress)
1.6. Start the mininet emulator with $sudo mn
1.7. This will create a default mininet topology (Fig. 1) with:
    a. 2 end hosts & 1 switch (data plane) and
    b. a default reference SDN controller (control plane)
1.8. Go to the mininet prompt and type help command to see all the available commands

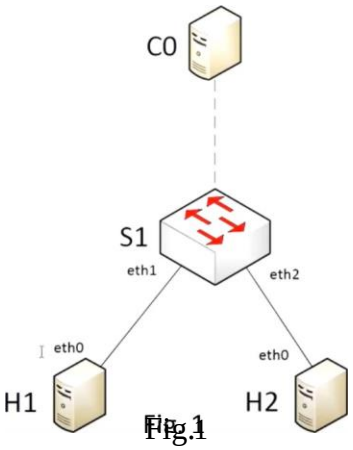

Fig.1

1.9. Test whether hl can ping h2 from the mininet prompt.
    a. Is the ping successful?
    b. How will the first packet from hl be handled by the switch sl?
    c. How is the first packet handled different from the traditional network?

a. Yes
b. The switch first adds the mac address of h1 to its forwarding table. In addition, since the switch initially did not know which port h2 is connected to, it duplicates the packets and sends them out as a broadcast, except the port that the packet was initially received on.
c. The first packet is sent out from the switch as a broadcast due to the mac address of h2 not being in the forwarding table, as compared to the later packets which are forwarded to h2 as h2's mac address is now present in the forwarding table of S1.

Answer 1.9 a, b & c in the below box [3 marks]

1.10. What are all the available nodes in the topology? Which command will you use to see all the details of the nodes in the network?
Include the screenshot(s) in the below box and describe [2 mark]

The available nodes are Host h1, Host h2, OVSSwitch s1 and Controller ce.

Command to be used is “dump”.

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1731>
<Host h2: h2-eth0:10.0.0.2 pid=1733>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=1738>
<Controller c0: 127.0.0.1:6653 pid=1724>
```

1.11. From the mininet prompt, type the command iperf and iperfudp to measure the tcp and udp bandwidth, respectively.

1.12. Quit from mininet and do a cleanup using the command sudo mn -c.

1.13. Write a python script to create a custom topology (custom_topo.py) as shown in Fig 2 (below) with 3 switches, and 2 end hosts in the /home/mininet/ folder and run using mininet (refer to sample script in /mininet/custom).

sudo mn --custom custom_topo.py --topo mytopo

(Note: For Fig.2 topology :wqthe eth_no. need not be the same)

1.14. Provide the python script used to create the topology in Fig.2 in the below box [6 marks]

```
from mininet.topo import Topo

class MyTopo( Topo ):

   "Simple topology example."

   def build( self ):

      "Create custom topo."

      # Add hosts and switches

      H1 = self.addHost( 'h1' )

      H2 = self.addHost( 'h2' )

      S1 = self.addSwitch( 's1' )

      S2 = self.addSwitch( 's2' )

      S3 = self.addSwitch( 's3' )

      # Add links

      self.addLink(H1, S2)

      self.addLink(H2, S3)

      self.addLink(S2, S3)

      self.addLink(S2, S1)

      self.addLink(S1, S3)

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

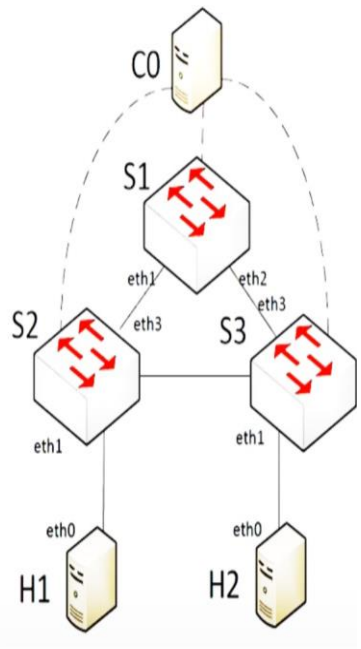

Fig. 2

1.15. On the mininet prompt ping from hl to h2.

(a) What output do you get?

Include the screenshot of the output below in the below box [1 marks]

Destination host unreachable.

```
mininet> H1 ping H2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable
From 10.0.0.1 icmp_seq=3 Destination Host Unreachable
From 10.0.0.1 icmp_seq=4 Destination Host Unreachable
```

(b) Run Wireshark/Tshark from the mininet vm and capture traffic from the 10 (loopback) interface. When you ping from hl to h2 what OpenFlow messages do you see? Include the screenshot of the output below in the below box [3 marks]

I see a lot of OpenFlow OFPT_PACKET_IN and OFPT_PACKET_OUT messages.

```
5105 8.077305234 9a:3e:d5:c0:05:69 ? Broadcast    OpenFlow 132 Type: OFPT_PACKET_OUT
5106 8.082957046 9a:3e:d5:c0:05:69 ? Broadcast    OpenFlow 132 Type: OFPT_PACKET_OUT
5107 8.082961074 9a:3e:d5:c0:05:69 ? Broadcast    OpenFlow 132 Type: OFPT_PACKET_OUT
5108 8.082963999 9a:3e:d5:c0:05:69 ? Broadcast    OpenFlow 132 Type: OFPT_PACKET_OUT
5109 8.082967726 9a:3e:d5:c0:05:69 ? Broadcast    OpenFlow 132 Type: OFPT_PACKET_OUT
5110 8.083037798 9a:3e:d5:c0:05:69 ? Broadcast    OpenFlow 132 Type: OFPT_PACKET_OUT
5111 8.084491259 9a:3e:d5:c0:05:69 ? Broadcast    OpenFlow 126 Type: OFPT_PACKET_IN
5112 8.084493614 9a:3e:d5:c0:05:69 ? Broadcast    OpenFlow 126 Type: OFPT_PACKET_IN
```

Part2 — Mininet with Remote the Controller Floodlight:

Floodlight is an OpenFlow enterprise-class SDN controller developed in Java. We will be using the Floodlight controller to connect to the hardware SDN switches (without mininet) in Lab 8.

The objective of Part 2 is to get familiarized with the Floodlight SDN controller.

2.1. Run sudo mn -c from the terminal to cleanup old configs

2.2. From the Mininet terminal, download dependencies for Floodlight master and above:
sudo apt-get install build-essential ant maven python-dev

2.3. To download and build Floodlight, git clone git://github.com/floodlight/floodlight.git

2.4. After step 2.3, installation is complete, follow the below steps to build the Floodlight controller and set root permissions.

$ cd floodlight
$ g it submodule init
$ g it submodule update
$ sudo ant

Ensure BUILD SUCCESSFUL, then enter:
$ sudo mkdir /var/lib/floodlight
$ sudo chmod 777 /var/lib/floodlight

2.5. Alternatively (instead of step 2.1 - 2.4), you can download the Floodlight VM that has a build in Floodlight+Mininet+Wireshark installation.
https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/8650780/Floodlight+VM
After installing the Floodlight VM, build and install Floodlight SDN controller java program (username: floodlight and password: floodlight) floodlight $ sudo ant

2.6. Start the Floodlight controller after installation in Step 2.4 or 2.5 floodlight
$ java —jar target/ floodlight. jar

2.7. Add the Floodlight controller as the remote controller when running Mininet for topology in Fig 2. For example:

Aininet@mininet-vm: $ sudo mn - -custom ./customtopo.py --topo mytopo port-6653 --switch ovsk, protocols=OpenF10w13

2.8. Is the remote controller added to the network emulator? How do you show this?
Include the relevant screenshot of the output below in the below box [3 marks]

Yes, it is added. We can show this through dump, which dumps all nodes and their infomartion.

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=3557>
<Host h2: h2-eth0:10.0.0.2 pid=3560>
<OVSSwitch{'protocols': 'OpenFlow13'} s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None
 pid=3566>
<OVSSwitch{'protocols': 'OpenFlow13'} s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None
,s2-eth3:None pid=3569>
<OVSSwitch{'protocols': 'OpenFlow13'} s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None
,s3-eth3:None pid=3572>
<RemoteController{'ip': '192.168.199.130', 'port': 6653} c0: 192.168.199.130:665
3 pid=3551>
```

2.9. Issue pingall command to from the mininet prompt. Does is ping successfully or fails? why?
Include the relevant screenshot of the output below in the below box [2 marks]

The pings all succeed. This could be because the floodlight controller issues lldp packets to see all the links between the switches.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

----------END OF ASSIGNMENT -

Estimated time to complete Assignment is 1-3 days based on familiarity with Linux & VMS

If you are familiar with Linux, you might as well try out the native installation of Mininet and Floodlight controller from the source below. http://mininet.org/download/_ https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343544/lnstallation+Guide

Reference Links:

1. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343544/lnstallation+Guide
2. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/p ages/8650780/F100dlight+VM
3. https://github.com/mininet/mininet/releases/
4. http://mininet.org/vm-setup-notes/
5. b.ttp.:LLmjnjneurgL
6. http://mininet.org/download/
7. http://mininet.org/walkthrough/#custom-topologies