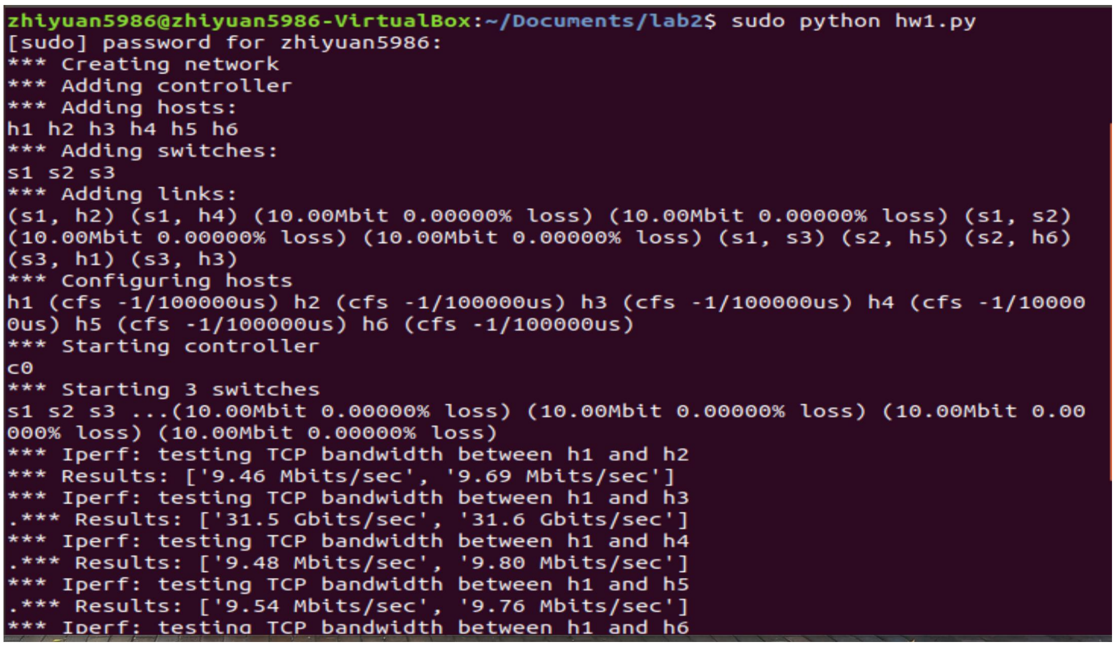
Lab2: Mininet

LIU Qiaoan 520030910220

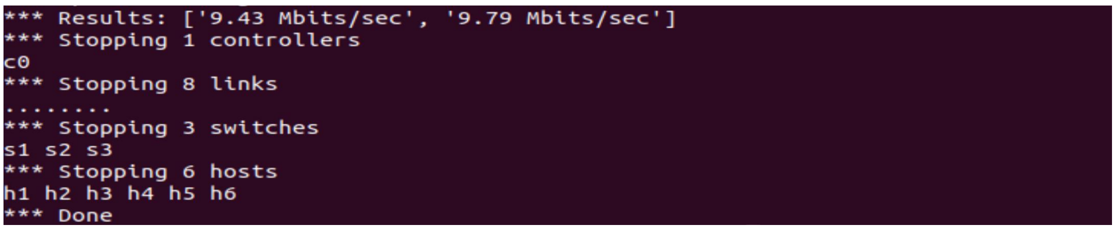
Summary

In this lab, I learn to use Mininet to simulate a network and test some basic command. Then I use ovs-ofctl command to adjust the rule of flows. My results are as follows:

0) First, I set up the environment. I clone Python code of Mininet from github, and then install OpenFlow. I test some examples and learn how to use ovs-ofctl commands.
1) In TASK 1, I use Python code to construct a topology with 3 switches and 6 hosts. We can see that the command of "iperf" in Python code print the throughput in the screen. The TCP throughput between h1 and h3 is much larger then others, since the bandwidth of (s1, s2) and (s2, s3) are both 10Mbps.



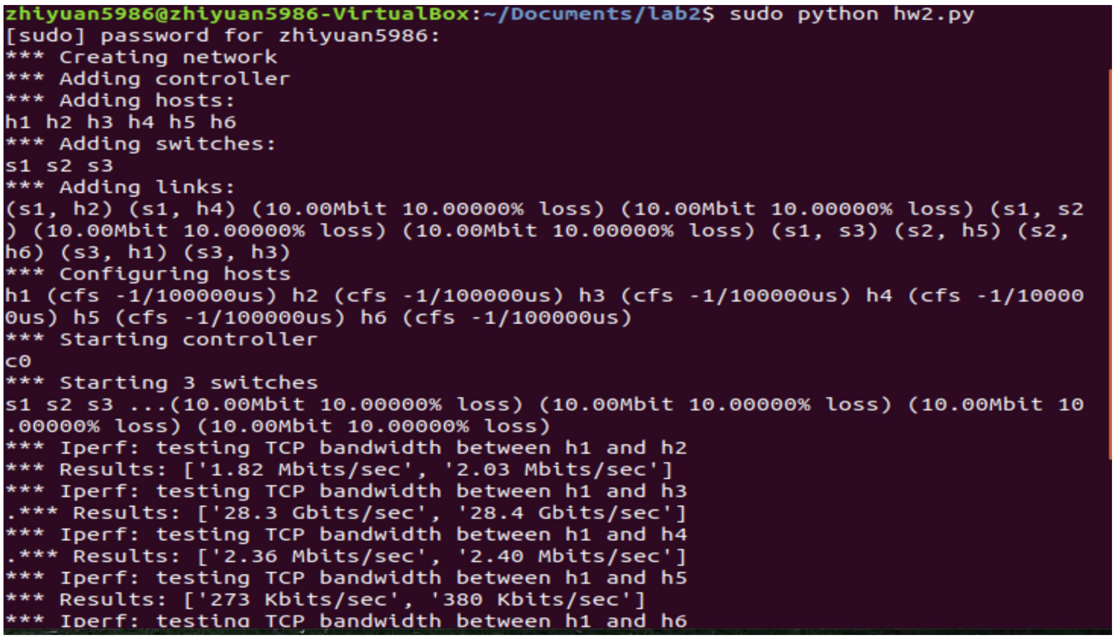
```
zhiyuan5986@zhiyuan5986-VirtualBox:~/Documents/lab2$ sudo python hw1.py
[sudo] password for zhiyuan5986:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, h2) (s1, h4) (10.00Mbit 0.00000% loss) (10.00Mbit 0.00000% loss) (s1, s2)
(10.00Mbit 0.00000% loss) (10.00Mbit 0.00000% loss) (s1, s3) (s2, h5) (s2, h6)
(s3, h1) (s3, h3)
*** Configuring hosts
h1 (cfs -1/100000us) h2 (cfs -1/100000us) h3 (cfs -1/100000us) h4 (cfs -1/10000
0us) h5 (cfs -1/100000us) h6 (cfs -1/100000us)
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...(10.00Mbit 0.00000% loss) (10.00Mbit 0.00000% loss) (10.00Mbit 0.00
000% loss) (10.00Mbit 0.00000% loss)
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['9.46 Mbits/sec', '9.69 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h3
.*** Results: ['31.5 Gbits/sec', '31.6 Gbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h4
.*** Results: ['9.48 Mbits/sec', '9.80 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h5
.*** Results: ['9.54 Mbits/sec', '9.76 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h6
```



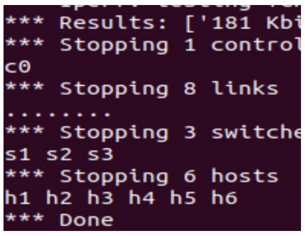
```
*** Results: ['9.43 Mbits/sec', '9.79 Mbits/sec']
*** Stopping 1 controllers
c0
*** Stopping 8 links
........
*** Stopping 3 switches
s1 s2 s3
*** Stopping 6 hosts
h1 h2 h3 h4 h5 h6
*** Done
```

Fig.1 Construct the topology and test throughput.

2) In TASK 2, I set the packet loss rate of the link (s1,s2) and (s1,s3) as 10%. Then I run the code to observe the output. As we can see, this time there are apparent drop of TCP throughput.



```
zhiyuan5986@zhiyuan5986-VirtualBox:~/Documents/lab2$ sudo python hw2.py
[sudo] password for zhiyuan5986:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, h2) (s1, h4) (10.00Mbit 10.00000% loss) (10.00Mbit 10.00000% loss) (s1, s2
) (10.00Mbit 10.00000% loss) (10.00Mbit 10.00000% loss) (s1, s3) (s2, h5) (s2,
h6) (s3, h1) (s3, h3)
*** Configuring hosts
h1 (cfs -1/100000us) h2 (cfs -1/100000us) h3 (cfs -1/100000us) h4 (cfs -1/10000
0us) h5 (cfs -1/100000us) h6 (cfs -1/100000us)
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...(10.00Mbit 10.00000% loss) (10.00Mbit 10.00000% loss) (10.00Mbit 10
.00000% loss) (10.00Mbit 10.00000% loss)
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['1.82 Mbits/sec', '2.03 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h3
.*** Results: ['28.3 Gbits/sec', '28.4 Gbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h4
.*** Results: ['2.36 Mbits/sec', '2.40 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h5
*** Results: ['273 Kbits/sec', '380 Kbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h6
```



```
*** Results: ['181 Kbi
*** Stopping 1 control
c0
*** Stopping 8 links
........
*** Stopping 3 switche
s1 s2 s3
*** Stopping 6 hosts
h1 h2 h3 h4 h5 h6
*** Done
```

Fig.2 Set the packet loss rate as 10%.

3) In TASK 3, I add another link between s2 and s3. I first use "pingall" to test the connectivity but they are all failed. That's because there is a loop, these packets will continue in the loop and never reach the hosts. When the switches are not configured, they will broadcast the packets and check the connectivity of near hosts. Then the packet will be transferred between switches again and again.

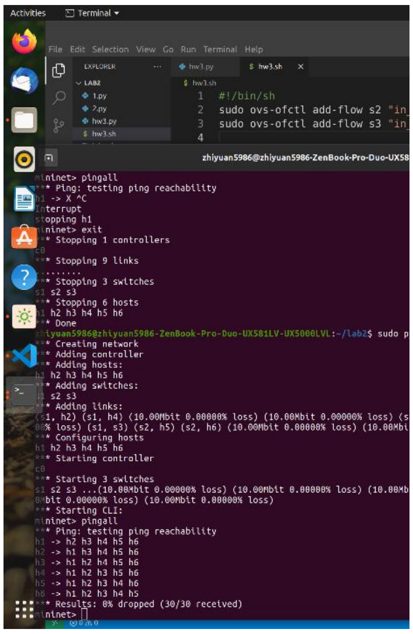
4) Then I use "net" to check the link information and write a sh file to add flow rules. My scheme is to add two flow rules on switch2 and switch 3. When a packet is transferred to switch 2 and switch 3, it will be transferred to hosts connected to them. This scheme can avoid looping endlessly. When a packet goes through the added link, it will be transmitted to the neighbor host instead of being broadcast in the loop.

   But, in my practice, I find some problems:

   (i). If I start the mininet, then run the sh file, and then pingall, we can see that all hosts are connected (Fig 3).

   (ii). If I start the mininet, then FIRST pingall and check the net information, then run sh file, and then pingall again. But we can see that in this time, some hosts are not connected

(Fig 4).



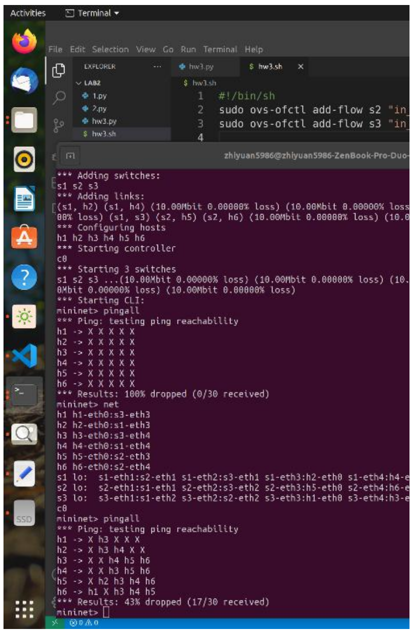
```
#!/bin/sh
sudo ovs-ofctl add-flow s2 "in_
sudo ovs-ofctl add-flow s3 "in_
```

```
zhiyuan5986@zhiyuan5986-ZenBook-Pro-Duo-UX58
mininet> pingall
*** Ping: testing ping reachability
h1 -> X ^C
Interrupt
stopping h1
mininet> exit
*** Stopping 1 controllers
c0

*** Stopping 9 links
.........
*** Stopping 3 switches
s1 s2 s3
*** Stopping 6 hosts
h1 h2 h3 h4 h5 h6
*** Done
zhiyuan5986@zhiyuan5986-ZenBook-Pro-Duo-UX581LV-UX5000LVL:~/lab2$ sudo p
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, h2) (s1, h4) (10.00Mbit 0.00000% loss) (10.00Mbit 0.00000% loss) (s
00% loss) (s1, s3) (s2, h5) (s2, h6) (10.00Mbit 0.00000% loss) (10.00Mbi
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0

*** Starting 3 switches
s1 s2 s3 ...(10.00Mbit 0.00000% loss) (10.00Mbit 0.00000% loss) (10.00Mb
0Mbit 0.00000% loss) (10.00Mbit 0.00000% loss)
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6
h2 -> h1 h3 h4 h5 h6
h3 -> h1 h2 h4 h5 h6
h4 -> h1 h2 h3 h5 h6
h5 -> h1 h2 h3 h4 h6
h6 -> h1 h2 h3 h4 h5
*** Results: 0% dropped (30/30 received)
mininet> 
```

Fig.3 sh->pingall.



```
#!/bin/sh
sudo ovs-ofctl add-flow s2 "in_
sudo ovs-ofctl add-flow s3 "in_
```

```
zhiyuan5986@zhiyuan5986-ZenBook-Pro-Duo-
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, h2) (s1, h4) (10.00Mbit 0.00000% loss) (10.00Mbit 0.00000% loss
00% loss) (s1, s3) (s2, h5) (s2, h6) (10.00Mbit 0.00000% loss) (10.0
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...(10.00Mbit 0.00000% loss) (10.00Mbit 0.00000% loss) (10.
0Mbit 0.00000% loss) (10.00Mbit 0.00000% loss)
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X X X
h2 -> X X X X X
h3 -> X X X X X
h4 -> X X X X X
h5 -> X X X X X
h6 -> X X X X X
*** Results: 100% dropped (0/30 received)
mininet> net
h1 h1-eth0:s3-eth3
h2 h2-eth0:s1-eth3
h3 h3-eth0:s3-eth4
h4 h4-eth0:s1-eth4
h5 h5-eth0:s2-eth3
h6 h6-eth0:s2-eth4
s1 lo:  s1-eth1:s2-eth1 s1-eth2:s3-eth1 s1-eth3:h2-eth0 s1-eth4:h4-e
s2 lo:  s2-eth1:s1-eth1 s2-eth2:s3-eth2 s2-eth3:h5-eth0 s2-eth4:h6-e
s3 lo:  s3-eth1:s1-eth2 s3-eth2:s2-eth2 s3-eth3:h1-eth0 s3-eth4:h3-e
c0
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 X X X
h2 -> X h3 h4 X X
h3 -> X X h4 h5 h6
h4 -> X X h3 h5 h6
h5 -> X h2 h3 h4 h6
h6 -> h1 X h3 h4 h5
*** Results: 43% dropped (17/30 received)
mininet> 
```

Fig.4 pingall->net->sh->pingall

I then communicate with TA for this problem.

Statement:

The last two figure are from my another Ubuntu 20.04 in my computer, since VirtualBox seems to be unstable when I do task 3.

Conclusion

In this lab, I study Mininet and ovs-ofctl command and research some knowledge of computer network.

Reference

[1] https://github.com/mininet/mininet

[2] https://github.com/mininet/mininet/wiki/Introduction-to-Mininet

[3] http://trainer.edu.mirantis.com/SDN50/ovs.html