

EE359 Computer Networks Lab Report 2

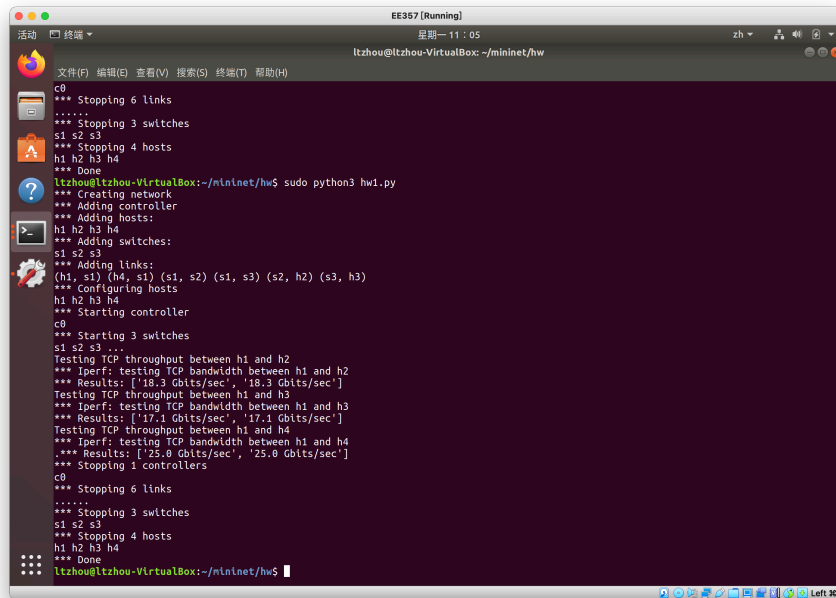
Zhou Litao 518030910407 F1803016

March 30, 2021, Spring Semester

Exercise 1 Simulate the following topology in Mininet. Set the link bandwidth for (s1,s2) and (s1,s3) as 10Mbps. Use iperf3 to test the TCP throughput between h1 to h2, h3, h4.(30 points)

Solution.

The TCP throughput between h1 to h2, h3, h4 is 18.3, 17.1 and 25.0 Gbits/sec respectively, shown in Figure 1.



```
EE357 [Running]
活动 终端
ltzhou@ltzhou-VirtualBox: ~/mininet/hw
*** Stopping 6 links
*** Stopping 3 switches
s1 s2 s3
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
ltzhou@ltzhou-VirtualBox:~/mininet/hw$ sudo python3 hw1.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h4, s1) (s1, s2) (s1, s3) (s2, h2) (s3, h3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
Testing TCP throughput between h1 and h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['18.3 Gbits/sec', '18.3 Gbits/sec']
Testing TCP throughput between h1 and h3
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['17.1 Gbits/sec', '17.1 Gbits/sec']
Testing TCP throughput between h1 and h4
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['25.0 Gbits/sec', '25.0 Gbits/sec']
*** Stopping 1 controllers
c0
*** Stopping 6 links
*** Stopping 3 switches
s1 s2 s3
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
ltzhou@ltzhou-VirtualBox:~/mininet/hw$
```

Figure 1: iperf3 test for the first exercise

Exercise 2 Now let us set the packet loss rate of the link (s1,s2) and (s1,s3) as 6%. Use iperf3 to test the TCP throughput between h1 to h2, h3, h4 again.(30 points)

Solution. The TCP throughput between h1 to h2, h3, h4 is 24.3, 19.4 and 16.3 Gbits/sec respectively, shown in Figure 2.

Exercise 3 Let us add another link between s2 and s3. Try to test the connectivity between all the hosts. What would happen? How would you solve the problem?(40 points)

Solution.

The process of creating this topology can be found in 5. The `pingall` command shows that no data can be transferred from one host to another.

This is because there exists a cycle of switches in the network, without any flooding control strategies. A **broadcasting storm** will happen. As broadcasts and multicasts are forwarded by switches out of every port, the switch or switches will repeatedly rebroadcast broadcast messages on the cycle and ultimately flood the network, causing no data can be actually transferred to the other host any more.

My solution in this problem is to disable one of the edge to eliminate the loop by using `ovs-ofctl` command. I proposed two solutions in solving this problem. Both solutions managed to eliminate the use of a link in the cycle either by dropping all packets received from the link or by not sending any packets to that link. See Figure 3 and Figure 4. Now the `pingall` can work again. See Figure 6

□

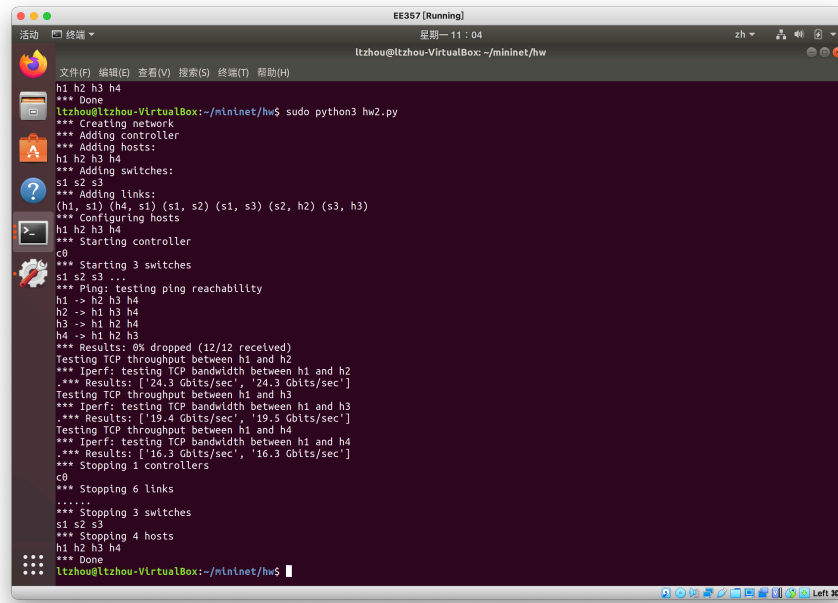


Figure 2: iperf3 test for the second exercise

Figure 3: Solution 1: disable all output to a link in the loop

```
sudo ovs-ofctl add-flow s1 "in_port=s1-eth4,actions=drop"
# s2 —x-> s1
sudo ovs-ofctl add-flow s2 "in_port=s2-eth1,actions=drop"
# s1 —x-> s2
```

Figure 4: Solution 2: disable all input to a link in the loop

```
sudo ovs-ofctl add-flow s2 "in_port=s2-eth1,actions=s2-eth2"
# s1 -> s2 -> h2
sudo ovs-ofctl add-flow s3 "in_port=s3-eth1,actions=s3-eth2"
# s1 -> s3 -> h3
sudo ovs-ofctl add-flow s2 "in_port=s2-eth2,actions=s2-eth1"
# h2 -> s2 -> s1
sudo ovs-ofctl add-flow s3 "in_port=s3-eth2,actions=s3-eth1"
# h3 -> s3 -> s1
```

```
EE357 [Running]
ltzhou@ltzhou-VirtualBox: ~/mininet
ltzhou@ltzhou-VirtualBox:~/mininet$ sudo python3 hw/hw3.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h4, s1) (s1, s2) (s1, s3) (s2, h2) (s2, s3) (s3, h3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
Dumping host connections
h1 h1-eth0:s1-eth3
h2 h2-eth0:s2-eth2
h3 h3-eth0:s3-eth2
h4 h4-eth0:s1-eth1
s1 lo: s1-eth1:h4-eth0 s1-eth2:s3-eth1 s1-eth3:h1-eth0 s1-eth4:s2-eth1
s2 lo: s2-eth1:s1-eth4 s2-eth2:h2-eth0 s2-eth3:s3-eth3
s3 lo: s3-eth1:s1-eth2 s3-eth2:h3-eth0 s3-eth3:s2-eth3
Ready To Ping
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X X X
h3 -> X X X
h4 -> X X X
*** Results: 100% dropped (0/12 received)
*** Stopping 1 controllers
c0
*** Stopping 7 links
.....
*** Stopping 3 switches
s1 s2 s3
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
ltzhou@ltzhou-VirtualBox:~/mininet$
```

Figure 5: pingall fails on loop switches

```
EE357 [Running]
星期六 23:10
ltzhou@ltzhou-VirtualBox: ~/mininet

*** Stopping 3 switches
s1 s2 s3
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
ltzhou@ltzhou-VirtualBox:~/mininet$ sudo python3 hw/hw3.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h4, s1) (s1, s2) (s1, s3) (s2, h2) (s2, s3) (s3, h3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
Dumping host connections
h1 h1-eth0:s1-eth3
h2 h2-eth0:s2-eth2
h3 h3-eth0:s3-eth2
h4 h4-eth0:s1-eth1
s1 lo: s1-eth1:h4-eth0 s1-eth2:s3-eth1 s1-eth3:h1-eth0 s1-eth4:s2-eth1
s2 lo: s2-eth1:s1-eth4 s2-eth2:h2-eth0 s2-eth3:s3-eth3
s3 lo: s3-eth1:s1-eth2 s3-eth2:h3-eth0 s3-eth3:s2-eth3
Ready To Ping
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
*** Stopping 1 controllers
c0
*** Stopping 7 links
.....
*** Stopping 3 switches
s1 s2 s3
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
ltzhou@ltzhou-VirtualBox:~/mininet$

ltzhou@ltzhou-VirtualBox:~/mininet/hw$ sh hw3.sh
Custom Flows Successfully Set
ltzhou@ltzhou-VirtualBox:~/mininet/hw$
```

Figure 6: pingall works again after flow-control