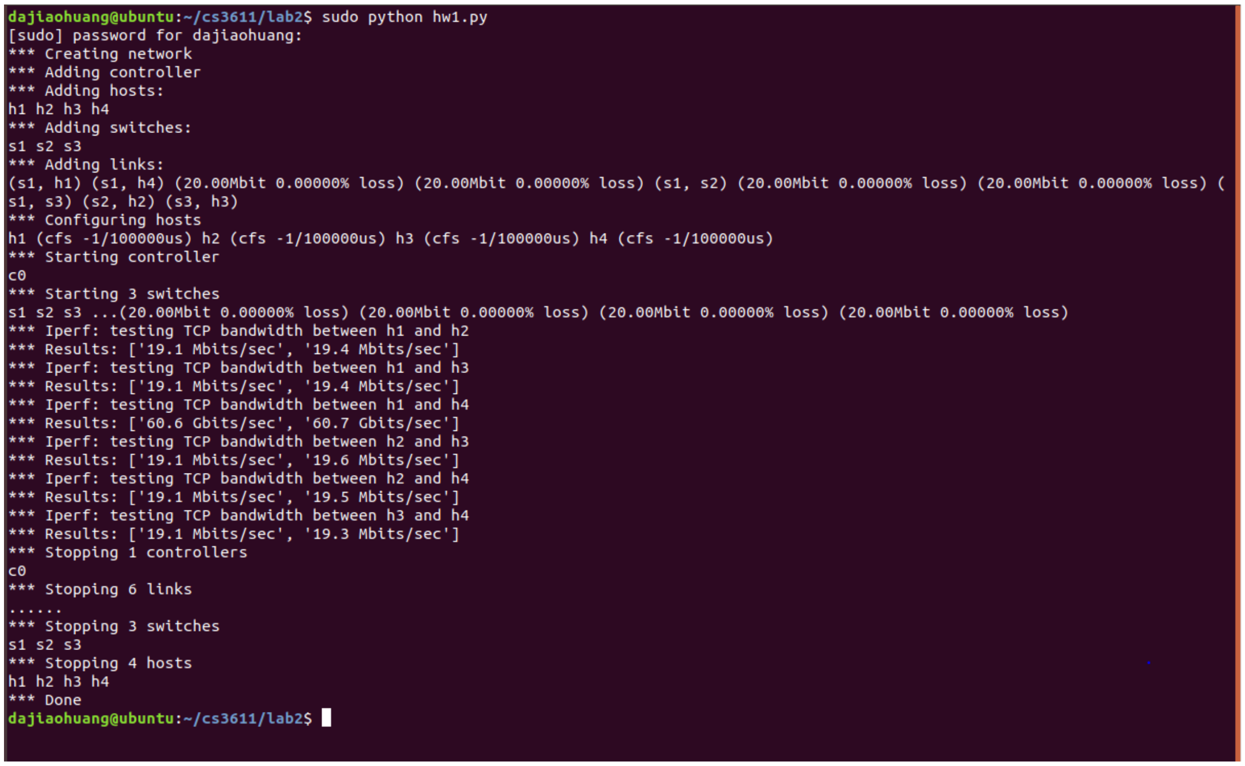
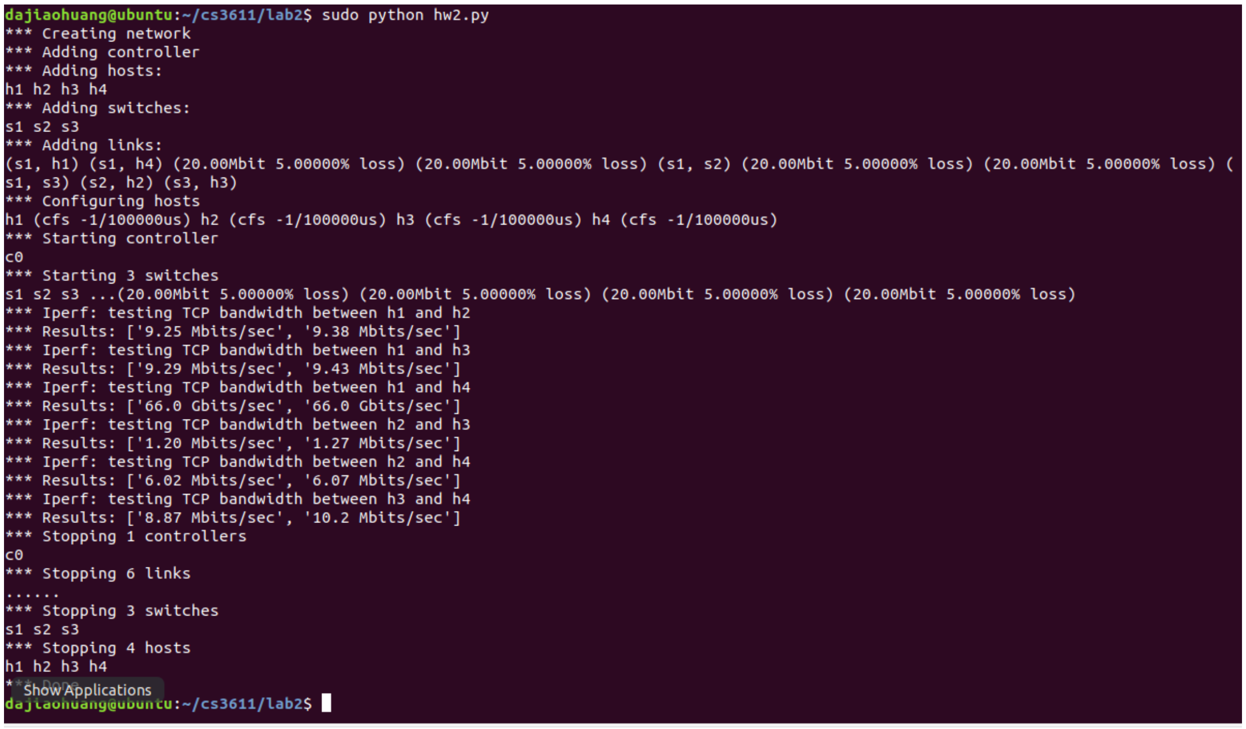
1.



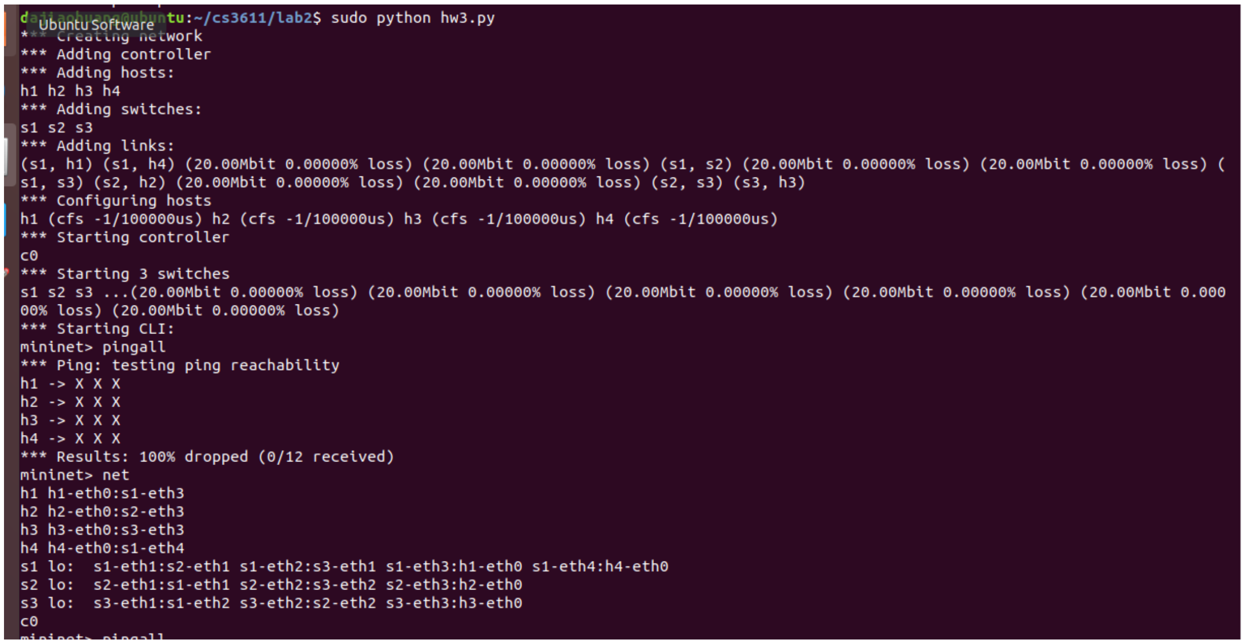
```
dajiaohuang@ubuntu:~/cs3611/lab2$ sudo python hw1.py
[sudo] password for dajiaohuang:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, h1) (s1, h4) (20.00Mbit 0.00000% loss) (20.00Mbit 0.00000% loss) (s1, s2) (20.00Mbit 0.00000% loss) (20.00Mbit 0.00000% loss) (
s1, s3) (s2, h2) (s3, h3)
*** Configuring hosts
h1 (cfs -1/100000us) h2 (cfs -1/100000us) h3 (cfs -1/100000us) h4 (cfs -1/100000us)
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...(20.00Mbit 0.00000% loss) (20.00Mbit 0.00000% loss) (20.00Mbit 0.00000% loss) (20.00Mbit 0.00000% loss)
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['19.1 Mbits/sec', '19.4 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['19.1 Mbits/sec', '19.4 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['60.6 Gbits/sec', '60.7 Gbits/sec']
*** Iperf: testing TCP bandwidth between h2 and h3
*** Results: ['19.1 Mbits/sec', '19.6 Mbits/sec']
*** Iperf: testing TCP bandwidth between h2 and h4
*** Results: ['19.1 Mbits/sec', '19.5 Mbits/sec']
*** Iperf: testing TCP bandwidth between h3 and h4
*** Results: ['19.1 Mbits/sec', '19.3 Mbits/sec']
*** Stopping 1 controllers
c0
*** Stopping 6 links
......
*** Stopping 3 switches
s1 s2 s3
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
dajiaohuang@ubuntu:~/cs3611/lab2$
```

2.



```
dajiaohuang@ubuntu:~/cs3611/lab2$ sudo python hw2.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, h1) (s1, h4) (20.00Mbit 5.00000% loss) (20.00Mbit 5.00000% loss) (s1, s2) (20.00Mbit 5.00000% loss) (20.00Mbit 5.00000% loss) (
s1, s3) (s2, h2) (s3, h3)
*** Configuring hosts
h1 (cfs -1/100000us) h2 (cfs -1/100000us) h3 (cfs -1/100000us) h4 (cfs -1/100000us)
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...(20.00Mbit 5.00000% loss) (20.00Mbit 5.00000% loss) (20.00Mbit 5.00000% loss) (20.00Mbit 5.00000% loss)
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['9.25 Mbits/sec', '9.38 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['9.29 Mbits/sec', '9.43 Mbits/sec']
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['66.0 Gbits/sec', '66.0 Gbits/sec']
*** Iperf: testing TCP bandwidth between h2 and h3
*** Results: ['1.20 Mbits/sec', '1.27 Mbits/sec']
*** Iperf: testing TCP bandwidth between h2 and h4
*** Results: ['6.02 Mbits/sec', '6.07 Mbits/sec']
*** Iperf: testing TCP bandwidth between h3 and h4
*** Results: ['8.87 Mbits/sec', '10.2 Mbits/sec']
*** Stopping 1 controllers
c0
*** Stopping 6 links
......
*** Stopping 3 switches
s1 s2 s3
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
dajiaohuang@ubuntu:~/cs3611/lab2$
```

3.



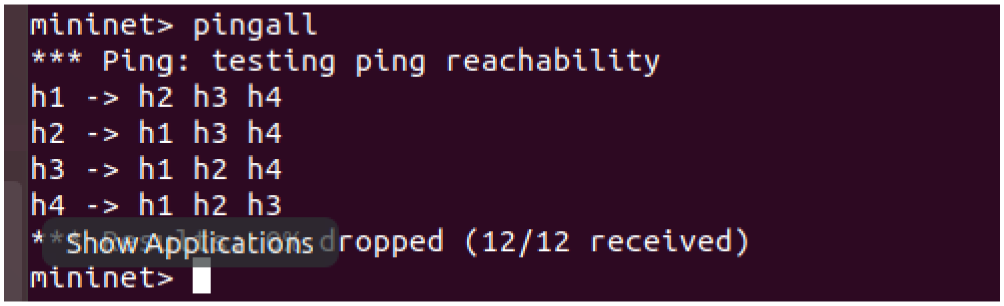
```
daijiaohuang@ubuntu:~/cs3611/lab2$ sudo python hw3.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, h1) (s1, h4) (20.00Mbit 0.00000% loss) (20.00Mbit 0.00000% loss) (s1, s2) (20.00Mbit 0.00000% loss) (20.00Mbit 0.00000% loss) (
s1, s3) (s2, h2) (20.00Mbit 0.00000% loss) (20.00Mbit 0.00000% loss) (s2, s3) (s3, h3)
*** Configuring hosts
h1 (cfs -1/100000us) h2 (cfs -1/100000us) h3 (cfs -1/100000us) h4 (cfs -1/100000us)
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...(20.00Mbit 0.00000% loss) (20.00Mbit 0.00000% loss) (20.00Mbit 0.00000% loss) (20.00Mbit 0.00000% loss) (20.00Mbit 0.000
00% loss) (20.00Mbit 0.00000% loss)
*** Starting CLI:
mininet> pingall
*** 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)
mininet> net
h1 h1-eth0:s1-eth3
h2 h2-eth0:s2-eth3
h3 h3-eth0:s3-eth3
h4 h4-eth0:s1-eth4
s1 lo:  s1-eth1:s2-eth1 s1-eth2:s3-eth1 s1-eth3:h1-eth0 s1-eth4:h4-eth0
s2 lo:  s2-eth1:s1-eth1 s2-eth2:s3-eth2 s2-eth3:h2-eth0
s3 lo:  s3-eth1:s1-eth2 s3-eth2:s2-eth2 s3-eth3:h3-eth0
c0
```

If no extra flow is applied, pingall will fail as the ping signal go into the loop.

From net command we can know the ports' id. Accorrding to the id, we can write such an sh, and add these flow to the switchers, so that every time the switchers are reached, we search for hosts first and the switchers.

```
#!/bin/sh
sudo ovs-ofctl add-flow s2 "in_port=s2-eth1,actions=output:s2-eth3,s2-eth2"
sudo ovs-ofctl add-flow s3 "in_port=s3-eth2,actions=output:s3-eth3,s3-eth1"
sudo ovs-ofctl add-flow s3 "in_port=s3-eth1,actions=output:s3-eth3,s3-eth2"
sudo ovs-ofctl add-flow s2 "in_port=s2-eth2,actions=output:s2-eth3,s2-eth1"
sudo ovs-ofctl add-flow s1 "in_port=s1-eth1,actions=output:s1-eth3,s1-eth4,s1-eth2"
sudo ovs-ofctl add-flow s1 "in_port=s1-eth2,actions=output:s1-eth3,s1-eth4,s1-eth1"
```

The sh specifies in what order we search the ports when the signal comes from a specific in-port, which is basically host first, switchers next, never go back.



```
mininet> pingall
*** 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)
mininet>
```

With this scheme, the ping-all succeeds.