EE 555 Final Project

Guangyu Chen 2916511097

Lu Xu 1788058427

To accomplish this project, we read tutorial in GitHub https://github.com/mininet/openflow-tutorial/wiki and some reference in GitHub such as https://github.com/hechengu/EE555, https://github.com/zhan849/ee555 and https://github.com/esha2008/SDN_firewall for firewall part.

After learning some basic steps about Mininet and POX, we finish part1, creating a learning switch and a router.

Learning swicth

First, we need open 2 SSH terminal through putty (windows) or termianl (Linux or Mac OS) and kill any running controller by the following command.

$ sudo killall controller

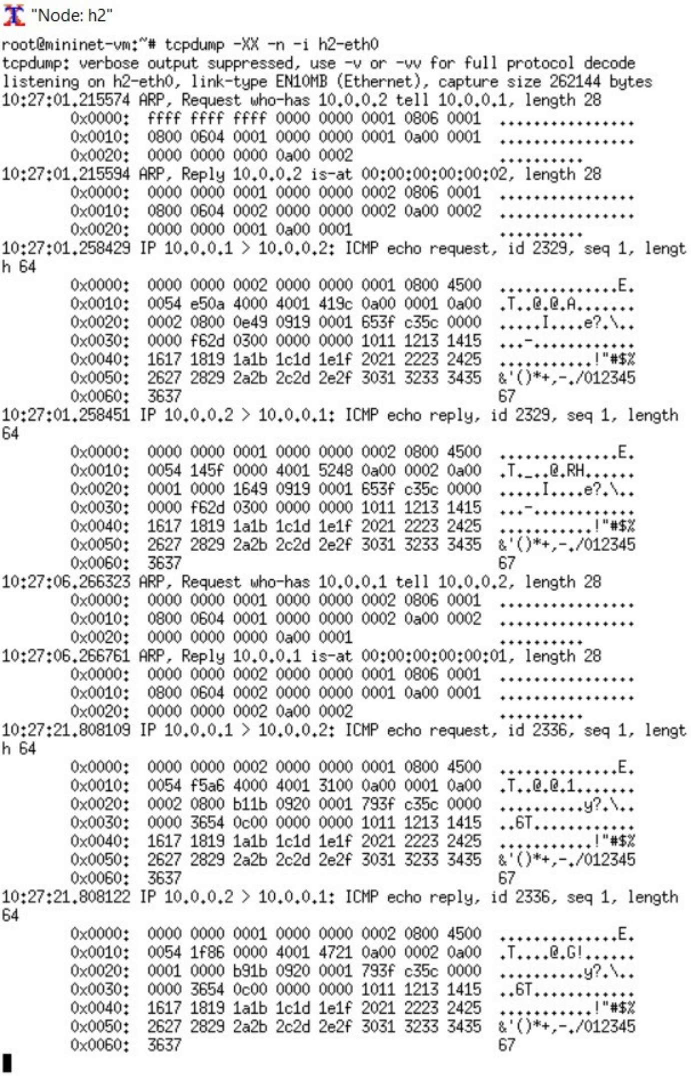
Also, run sudo mn -c to make sure that everything is "clean". Run the following command to initiate the topology.

$ sudo mn --topo single,3 --mac --switch ovsk --controller remote,ip=127.0.0.1,port=6633

Open another SSH termianl and run the following command to start controller.

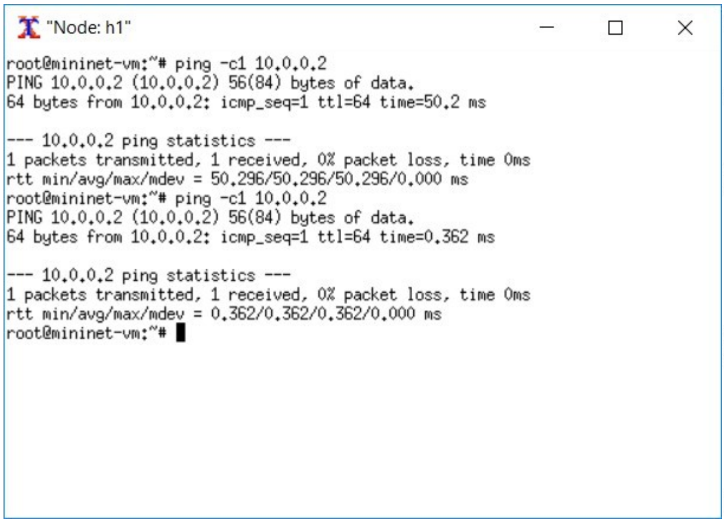
$ ./pox.py log.level --DEBUG misc.of_tutorial

Then, use xterm h1 h2 h3 to run 3 hosts seperately. If we let host2 and host3 listen and h1 ping host2 for twice, as a switch, it would only forward ARP request to host3 only once. For the second ping, switch should only forward it to host2. From the controller debug, we should only see Flooding for once.



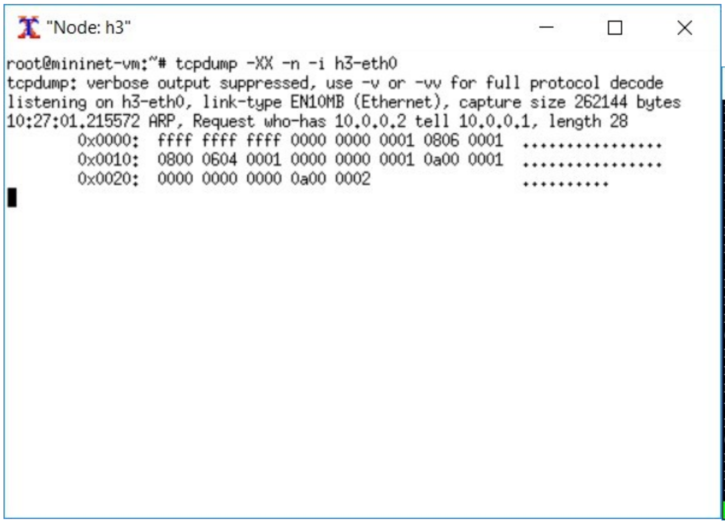
"Node: h2"

```
root@mininet-vm:~# tcpdump -XX -n -i h2-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h2-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:27:01.215574 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
        0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
        0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
        0x0020:  0000 0000 0000 0a00 0002                 ..........
10:27:01.215594 ARP, Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
        0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
        0x0010:  0800 0604 0002 0000 0000 0002 0a00 0002  ................
        0x0020:  0000 0000 0001 0a00 0001                 ..........
10:27:01.258429 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 2329, seq 1, lengt
h 64
        0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
        0x0010:  0054 e50a 4000 4001 419c 0a00 0001 0a00  .T..@.@.A.......
        0x0020:  0002 0800 0e49 0919 0001 653f c35c 0000  .....I....e?.\..
        0x0030:  0000 f62d 0300 0000 0000 1011 1213 1415  ...-............
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
10:27:01.258451 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 2329, seq 1, length
64
        0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
        0x0010:  0054 145f 0000 4001 5248 0a00 0002 0a00  .T._..@.RH......
        0x0020:  0001 0000 1649 0919 0001 653f c35c 0000  .....I....e?.\..
        0x0030:  0000 f62d 0300 0000 0000 1011 1213 1415  ...-............
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
10:27:06.266323 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
        0x0000:  0000 0000 0001 0000 0000 0002 0806 0001  ................
        0x0010:  0800 0604 0001 0000 0000 0002 0a00 0002  ................
        0x0020:  0000 0000 0000 0a00 0001                 ..........
10:27:06.266761 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
        0x0000:  0000 0000 0002 0000 0000 0001 0806 0001  ................
        0x0010:  0800 0604 0002 0000 0000 0001 0a00 0001  ................
        0x0020:  0000 0000 0002 0a00 0002                 ..........
10:27:21.808109 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 2336, seq 1, lengt
h 64
        0x0000:  0000 0000 0002 0000 0000 0001 0800 4500  ..............E.
        0x0010:  0054 f5a6 4000 4001 3100 0a00 0001 0a00  .T..@.@.1.......
        0x0020:  0002 0800 b11b 0920 0001 793f c35c 0000  ..........y?.\..
        0x0030:  0000 3654 0c00 0000 0000 1011 1213 1415  ..6T............
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
10:27:21.808122 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 2336, seq 1, length
64
        0x0000:  0000 0000 0001 0000 0000 0002 0800 4500  ..............E.
        0x0010:  0054 1f86 0000 4001 4721 0a00 0002 0a00  .T....@.G!......
        0x0020:  0001 0000 b91b 0920 0001 793f c35c 0000  ..........y?.\..
        0x0030:  0000 3654 0c00 0000 0000 1011 1213 1415  ..6T............
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
█
```

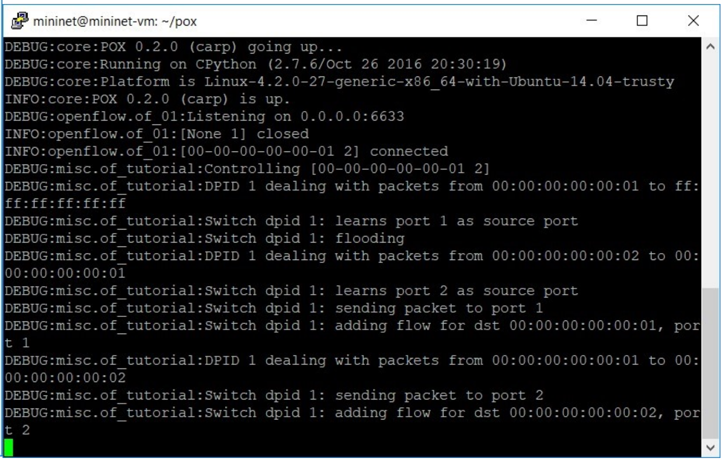


"Node: h1"

```
root@mininet-vm:~# ping -c1 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=50.2 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 50.296/50.296/50.296/0.000 ms
root@mininet-vm:~# ping -c1 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.362 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.362/0.362/0.362/0.000 ms
root@mininet-vm:~# █
```

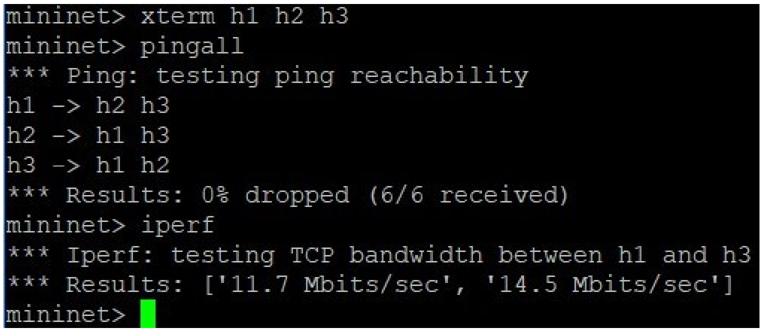


```
root@mininet-vm:~# tcpdump -XX -n -i h3-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h3-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:27:01.215572 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
        0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
        0x0010:  0800 0604 0001 0000 0000 0001 0a00 0001  ................
        0x0020:  0000 0000 0000 0a00 0002                 ..........
```



```
DEBUG:core:POX 0.2.0 (carp) going up...
DEBUG:core:Running on CPython (2.7.6/Oct 26 2016 20:30:19)
DEBUG:core:Platform is Linux-4.2.0-27-generic-x86_64-with-Ubuntu-14.04-trusty
INFO:core:POX 0.2.0 (carp) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[None 1] closed
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
DEBUG:misc.of_tutorial:Controlling [00-00-00-00-00-01 2]
DEBUG:misc.of_tutorial:DPID 1 dealing with packets from 00:00:00:00:00:01 to ff:
ff:ff:ff:ff:ff
DEBUG:misc.of_tutorial:Switch dpid 1: learns port 1 as source port
DEBUG:misc.of_tutorial:Switch dpid 1: flooding
DEBUG:misc.of_tutorial:DPID 1 dealing with packets from 00:00:00:00:00:02 to 00:
00:00:00:00:01
DEBUG:misc.of_tutorial:Switch dpid 1: learns port 2 as source port
DEBUG:misc.of_tutorial:Switch dpid 1: sending packet to port 1
DEBUG:misc.of_tutorial:Switch dpid 1: adding flow for dst 00:00:00:00:00:01, por
t 1
DEBUG:misc.of_tutorial:DPID 1 dealing with packets from 00:00:00:00:00:01 to 00:
00:00:00:00:02
DEBUG:misc.of_tutorial:Switch dpid 1: sending packet to port 2
DEBUG:misc.of_tutorial:Switch dpid 1: adding flow for dst 00:00:00:00:00:02, por
t 2
```

After verifying switch behaviour, we can test if switch can connect all nodes and rate for it. We find transmission rate for switch is significiantly faster than hub.



```
mininet> xterm h1 h2 h3
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['11.7 Mbits/sec', '14.5 Mbits/sec']
mininet>
```

Router exercise

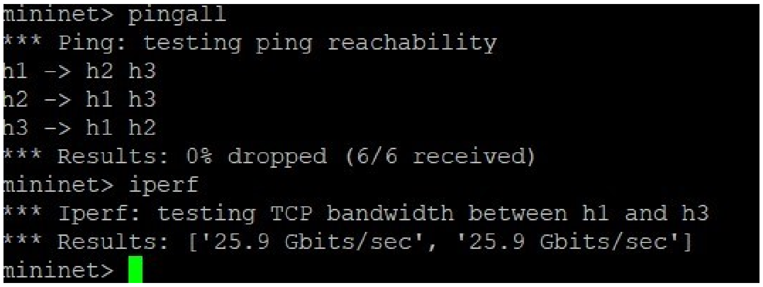
In this part, we build a different topology by the following command.

$ sudo mn --custom part1_topo.py --topo part1_topo --mac --controller=remote,ip=127.0.0.1,port=6633

In the other termianl, run controller by entering pox and enetering the following command.

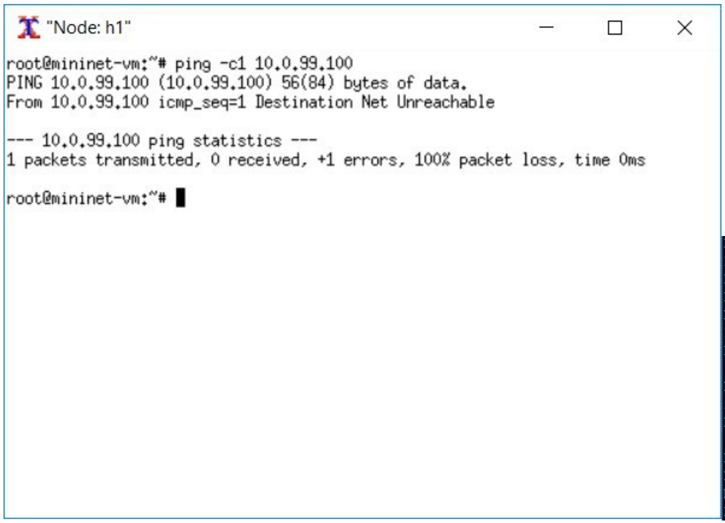
$ sudo ./pox.py log.level --DEBUG misc.part1_router misc.full_payload

Test the router by pingall and iperf, here is the result.

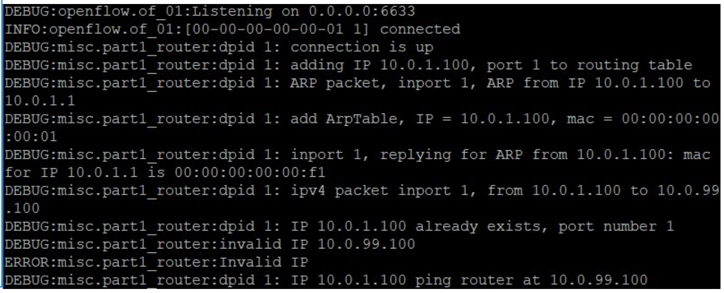


```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['25.9 Gbits/sec', '25.9 Gbits/sec']
mininet>
```

Test if router can yield ICMP destination unreachable message when we ping an unkown destination such as 10.0.99.100

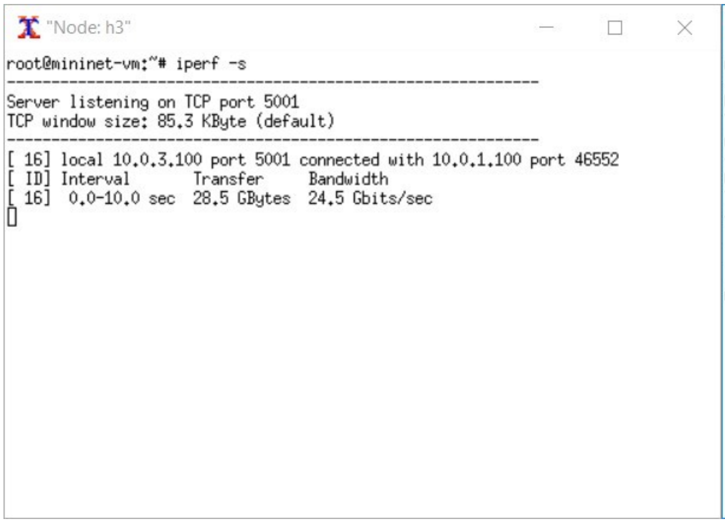


```
root@mininet-vm:~# ping -c1 10.0.99.100
PING 10.0.99.100 (10.0.99.100) 56(84) bytes of data.
From 10.0.99.100 icmp_seq=1 Destination Net Unreachable

--- 10.0.99.100 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

root@mininet-vm:~# 
```

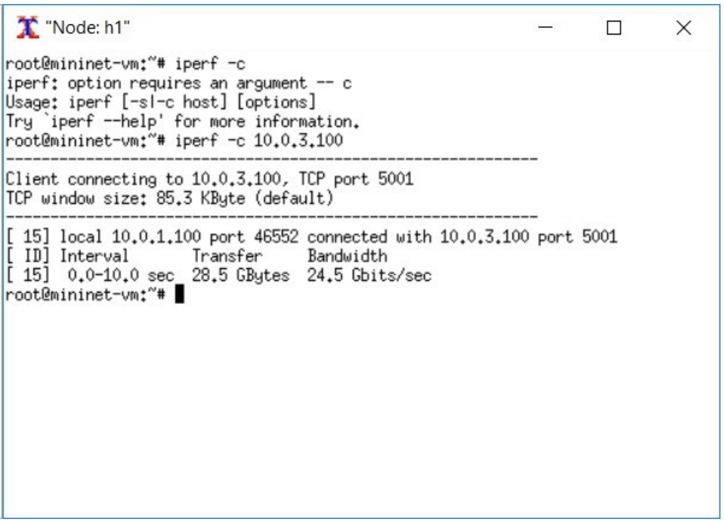


```
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
DEBUG:misc.part1_router:dpid 1: connection is up
DEBUG:misc.part1_router:dpid 1: adding IP 10.0.1.100, port 1 to routing table
DEBUG:misc.part1_router:dpid 1: ARP packet, inport 1, ARP from IP 10.0.1.100 to
10.0.1.1
DEBUG:misc.part1_router:dpid 1: add ArpTable, IP = 10.0.1.100, mac = 00:00:00:00
:00:01
DEBUG:misc.part1_router:dpid 1: inport 1, replying for ARP from 10.0.1.100: mac
for IP 10.0.1.1 is 00:00:00:00:00:f1
DEBUG:misc.part1_router:dpid 1: ipv4 packet inport 1, from 10.0.1.100 to 10.0.99
.100
DEBUG:misc.part1_router:dpid 1: IP 10.0.1.100 already exists, port number 1
DEBUG:misc.part1_router:invalid IP 10.0.99.100
ERROR:misc.part1_router:Invalid IP
DEBUG:misc.part1_router:dpid 1: IP 10.0.1.100 ping router at 10.0.99.100
```

Use host3 as iperf server and host1 as client, run iperf to test tcp and udp traffic.



```
root@mininet-vm:~# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 16] local 10.0.3.100 port 5001 connected with 10.0.1.100 port 46552
[ ID] Interval       Transfer     Bandwidth
[ 16]  0.0-10.0 sec  28.5 GBytes  24.5 Gbits/sec
```



```
root@mininet-vm:~# iperf -c
iperf: option requires an argument -- c
Usage: iperf [-s|-c host] [options]
Try `iperf --help' for more information.
root@mininet-vm:~# iperf -c 10.0.3.100
------------------------------------------------------------
Client connecting to 10.0.3.100, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 15] local 10.0.1.100 port 46552 connected with 10.0.3.100 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 15]  0.0-10.0 sec  28.5 GBytes  24.5 Gbits/sec
root@mininet-vm:~# 
```

Advanced Topology

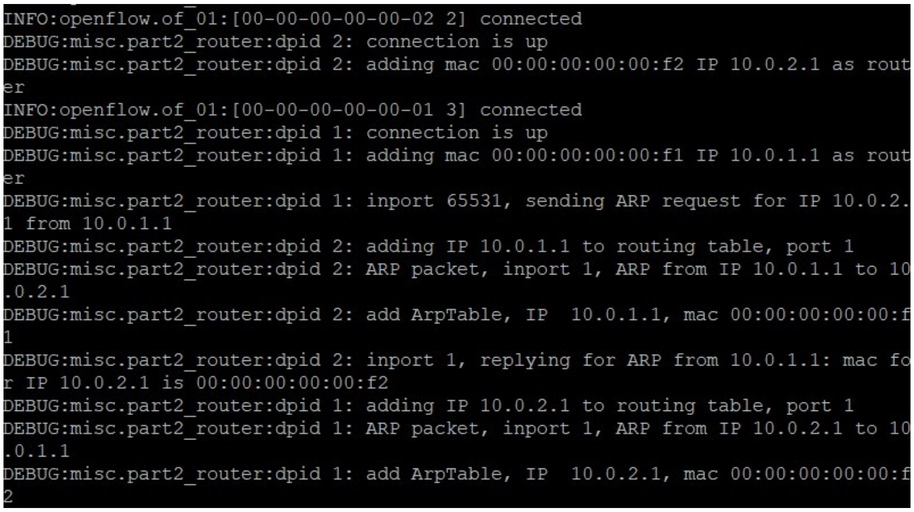
In this part, we build a advanced topology by the following command.

```
$ sudo mn --custom part2_topo.py --topo part2_topo --mac --controller=remote,ip=127.0.0.1,port=6633
```

In the other termianl, run controller by entering pox and enetering the following command.

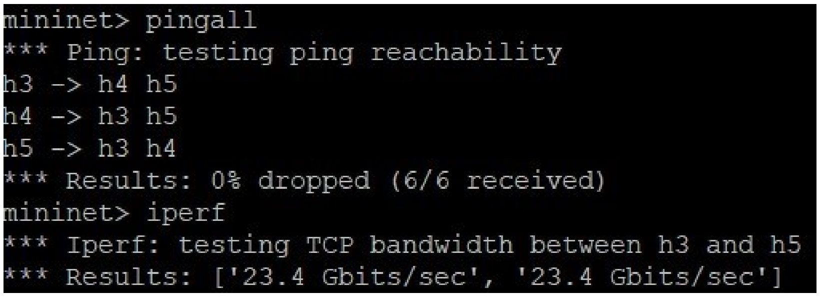
```
$ sudo ./pox.py log.level --DEBUG misc.part2_router misc.full_payload
```

When the controller boots up, two routers will communicate with each other.



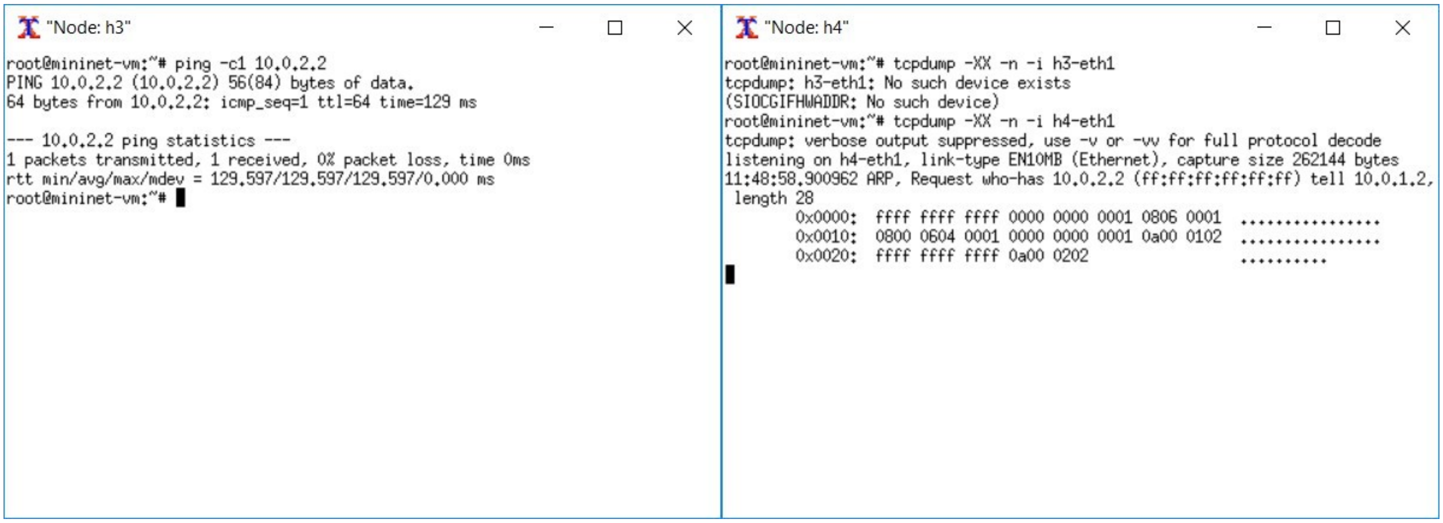
```
INFO:openflow.of_01:[00-00-00-00-00-02 2] connected
DEBUG:misc.part2_router:dpid 2: connection is up
DEBUG:misc.part2_router:dpid 2: adding mac 00:00:00:00:00:f2 IP 10.0.2.1 as rout
er
INFO:openflow.of_01:[00-00-00-00-00-01 3] connected
DEBUG:misc.part2_router:dpid 1: connection is up
DEBUG:misc.part2_router:dpid 1: adding mac 00:00:00:00:00:f1 IP 10.0.1.1 as rout
er
DEBUG:misc.part2_router:dpid 1: inport 65531, sending ARP request for IP 10.0.2.
1 from 10.0.1.1
DEBUG:misc.part2_router:dpid 2: adding IP 10.0.1.1 to routing table, port 1
DEBUG:misc.part2_router:dpid 2: ARP packet, inport 1, ARP from IP 10.0.1.1 to 10
.0.2.1
DEBUG:misc.part2_router:dpid 2: add ArpTable, IP  10.0.1.1, mac 00:00:00:00:00:f
1
DEBUG:misc.part2_router:dpid 2: inport 1, replying for ARP from 10.0.1.1: mac fo
r IP 10.0.2.1 is 00:00:00:00:00:f2
DEBUG:misc.part2_router:dpid 1: adding IP 10.0.2.1 to routing table, port 1
DEBUG:misc.part2_router:dpid 1: ARP packet, inport 1, ARP from IP 10.0.2.1 to 10
.0.1.1
DEBUG:misc.part2_router:dpid 1: add ArpTable, IP  10.0.2.1, mac 00:00:00:00:00:f
2
```

Test the router by pingall and iperf, here is the result.



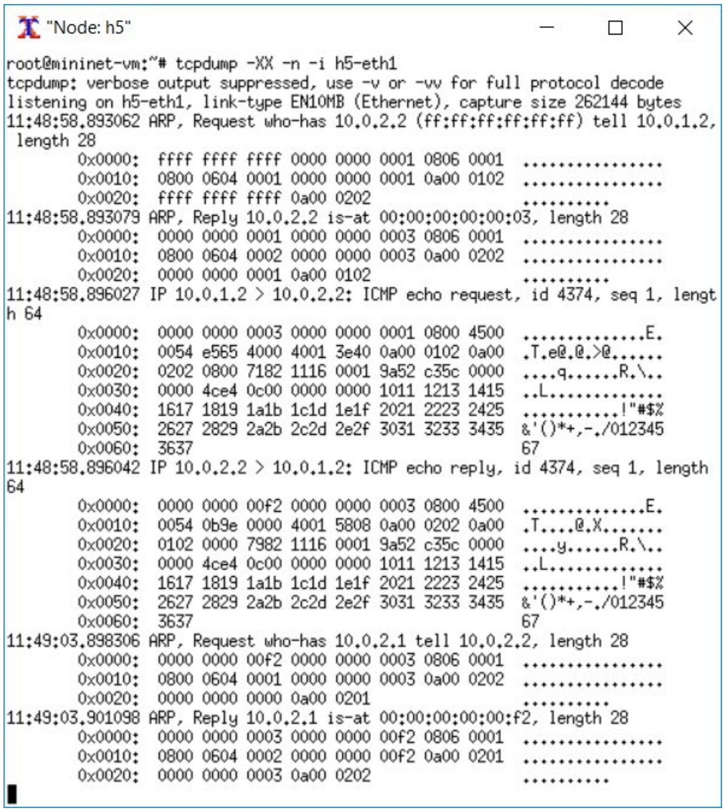
```
mininet> pingall
*** Ping: testing ping reachability
h3 -> h4 h5
h4 -> h3 h5
h5 -> h3 h4
*** Results: 0% dropped (6/6 received)
mininet> iperf
*** Iperf: testing TCP bandwidth between h3 and h5
*** Results: ['23.4 Gbits/sec', '23.4 Gbits/sec']
```

If we use xterm h3 h4 h5 and ping host5 on terminal of host3. Host4 will get an ARP request and ignore it, host 5 will receive ICMP request from router2 and reply it. In the pox termianl, router will generate corresponding message for it.



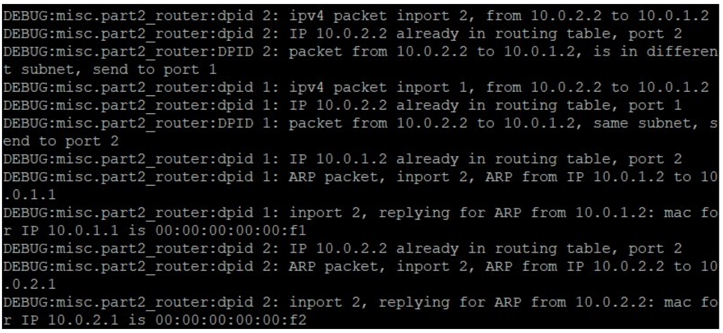
```
root@mininet-vm:~# ping -c1 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=64 time=129 ms

--- 10.0.2.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 129.597/129.597/129.597/0.000 ms
root@mininet-vm:~#
```

```
root@mininet-vm:~# tcpdump -XX -n -i h3-eth1
tcpdump: h3-eth1: No such device exists
(SIOCGIFHWADDR: No such device)
root@mininet-vm:~# tcpdump -XX -n -i h4-eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h4-eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
11:48:58.900962 ARP, Request who-has 10.0.2.2 (ff:ff:ff:ff:ff:ff) tell 10.0.1.2,
 length 28
        0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
        0x0010:  0800 0604 0001 0000 0000 0001 0a00 0102  ................
        0x0020:  ffff ffff ffff 0a00 0202                 ..........
```



"Node: h5"

```
root@mininet-vm:~# tcpdump -XX -n -i h5-eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h5-eth1, link-type EN10MB (Ethernet), capture size 262144 bytes
11:48:58.893062 ARP, Request who-has 10.0.2.2 (ff:ff:ff:ff:ff:ff) tell 10.0.1.2,
 length 28
        0x0000:  ffff ffff ffff 0000 0000 0001 0806 0001  ................
        0x0010:  0800 0604 0001 0000 0000 0001 0a00 0102  ................
        0x0020:  ffff ffff ffff 0a00 0202                 ..........
11:48:58.893079 ARP, Reply 10.0.2.2 is-at 00:00:00:00:00:03, length 28
        0x0000:  0000 0000 0001 0000 0000 0003 0806 0001  ................
        0x0010:  0800 0604 0002 0000 0000 0003 0a00 0202  ................
        0x0020:  0000 0000 0001 0a00 0102                 ..........
11:48:58.896027 IP 10.0.1.2 > 10.0.2.2: ICMP echo request, id 4374, seq 1, lengt
h 64
        0x0000:  0000 0000 0003 0000 0000 0001 0800 4500  ..............E.
        0x0010:  0054 e565 4000 4001 3e40 0a00 0102 0a00  .T.e@.@.>@......
        0x0020:  0202 0800 7182 1116 0001 9a52 c35c 0000  ....q......R.\..
        0x0030:  0000 4ce4 0c00 0000 0000 1011 1213 1415  ..L.............
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
11:48:58.896042 IP 10.0.2.2 > 10.0.1.2: ICMP echo reply, id 4374, seq 1, length
64
        0x0000:  0000 0000 00f2 0000 0000 0003 0800 4500  ..............E.
        0x0010:  0054 0b9e 0000 4001 5808 0a00 0202 0a00  .T....@.X.......
        0x0020:  0102 0000 7982 1116 0001 9a52 c35c 0000  ....y......R.\..
        0x0030:  0000 4ce4 0c00 0000 0000 1011 1213 1415  ..L.............
        0x0040:  1617 1819 1a1b 1c1d 1e1f 2021 2223 2425  ...........!"#$%
        0x0050:  2627 2829 2a2b 2c2d 2e2f 3031 3233 3435  &'()*+,-./012345
        0x0060:  3637                                     67
11:49:03.898306 ARP, Request who-has 10.0.2.1 tell 10.0.2.2, length 28
        0x0000:  0000 0000 00f2 0000 0000 0003 0806 0001  ................
        0x0010:  0800 0604 0001 0000 0000 0003 0a00 0202  ................
        0x0020:  0000 0000 0000 0a00 0201                 ..........
11:49:03.901098 ARP, Reply 10.0.2.1 is-at 00:00:00:00:00:f2, length 28
        0x0000:  0000 0000 0003 0000 0000 00f2 0806 0001  ................
        0x0010:  0800 0604 0002 0000 0000 00f2 0a00 0201  ................
        0x0020:  0000 0000 0003 0a00 0202                 ..........
```



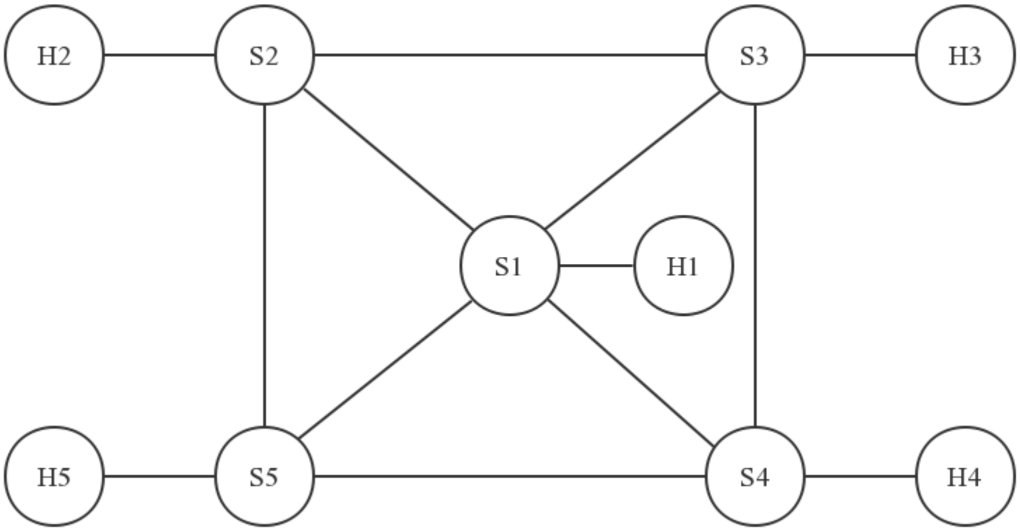
```
DEBUG:misc.part2_router:dpid 2: ipv4 packet inport 2, from 10.0.2.2 to 10.0.1.2
DEBUG:misc.part2_router:dpid 2: IP 10.0.2.2 already in routing table, port 2
DEBUG:misc.part2_router:DPID 2: packet from 10.0.2.2 to 10.0.1.2, is in differen
t subnet, send to port 1
DEBUG:misc.part2_router:dpid 1: ipv4 packet inport 1, from 10.0.2.2 to 10.0.1.2
DEBUG:misc.part2_router:dpid 1: IP 10.0.2.2 already in routing table, port 1
DEBUG:misc.part2_router:DPID 1: packet from 10.0.2.2 to 10.0.1.2, same subnet, s
end to port 2
DEBUG:misc.part2_router:dpid 1: IP 10.0.1.2 already in routing table, port 2
DEBUG:misc.part2_router:dpid 1: ARP packet, inport 2, ARP from IP 10.0.1.2 to 10
.0.1.1
DEBUG:misc.part2_router:dpid 1: inport 2, replying for ARP from 10.0.1.2: mac fo
r IP 10.0.1.1 is 00:00:00:00:00:f1
DEBUG:misc.part2_router:dpid 2: IP 10.0.2.2 already in routing table, port 2
DEBUG:misc.part2_router:dpid 2: ARP packet, inport 2, ARP from IP 10.0.2.2 to 10
.0.2.1
DEBUG:misc.part2_router:dpid 2: inport 2, replying for ARP from 10.0.2.2: mac fo
r IP 10.0.2.1 is 00:00:00:00:00:f2
```

Bonus

For the first part of bonus, we only create the topology but not finish the controller part.

For the second part of bonus, we use https://github.com/esha2008/SDN_firewall as reference to finish the function of firewall.

In this part, we build the topology like this



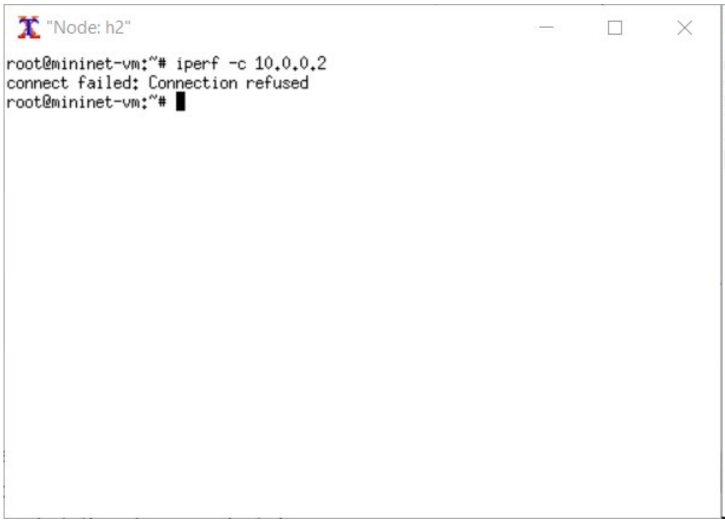
Open two SSH terminal for Mininet, in the first one, type:

```
$ sudo mn --custom firewall_topo.py --topo firewall_topo --mac --controller=remote,ip=127.0.0.1,port=6633
```

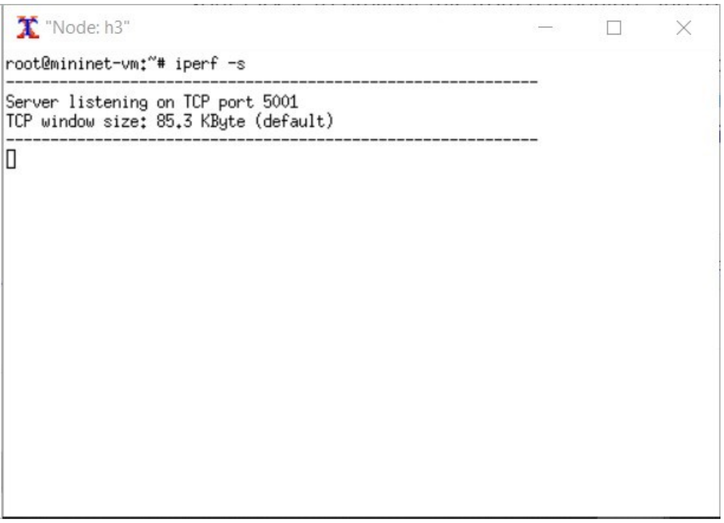
In the second terminal, type:

```
$ sudo ./pox.py forwarding.l2_learning openflow.discovery openflow.spanning_tree --no-flood --hold-down pox.misc.firewall
```

We set up a firewall between host2 and host3. If it works, when we set host3 as sever and host2 as client, host2 cannot send TCP fragment to host3.

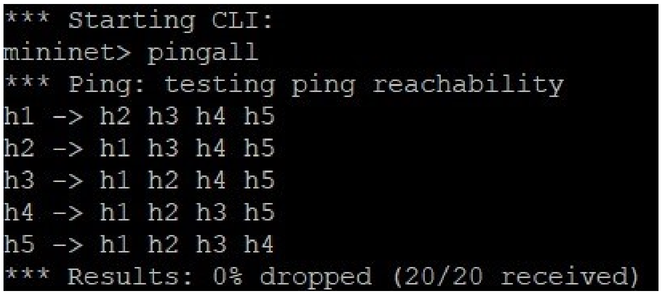


```
root@mininet-vm:~# iperf -c 10.0.0.2
connect failed: Connection refused
root@mininet-vm:~# 
```



```
root@mininet-vm:~# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

However, we can still use pingall to test if all 5 hosts are connected.



```
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
```