## Simple Datacenter topology (45 points):

```
sdn@sdn-course:~/mininet/custom$ sudo python3 SimpleDataTopology.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
a1 a2 c1 e1 e2 e3 e4
*** Adding links:
(a1, e1) (a1, e2) (a2, e3) (a2, e4) (c1, a1) (c1, a2) (e1, h1) (e1, h2) (e2, h3) (e2, h4) (e3, h5
) (e3, h6) (e4, h7) (e4, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
a1 a2 c1 e1 e2 e3 e4 ...
*** Starting CLI:
```

Above figure is the initiation of the topo.

Q1. What is the output for “pingall” command. (Please add a screenshot here.) (5 points)

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
```

Each host ping the other hosts successfully. Pingall shows the connectivity of all hosts by sending ICMP ping.

Q2. What is the output of “iperf” command. (Please add a screenshot here.) (5 points)

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h8
*** Results: ['64.7 Gbits/sec', '64.8 Gbits/sec']
mininet>
```

The iperf shows the quality of bandwidth between hosts.

# Fat-tree topology (55 points):

Initialization of Fat-tree:

```
*** Creating network
*** Adding controller
*** Adding hosts:
h0.0.2 h0.0.3 h0.1.2 h0.1.3 h1.0.2 h1.0.3 h1.1.2 h1.1.3 h2.0.2 h2.0.3 h2.1.2 h2.1.3 h3.0.2 h3.0.3 h3.1.2 h3.1.3
*** Adding switches:
agg02 agg03 agg12 agg13 agg22 agg23 agg32 agg33 cor11 cor12 cor21 cor22 edg00 edg01 edg10 edg11 edg20 edg21 edg30 edg31
*** Adding links:
(agg02, edg00) (agg02, edg01) (agg03, edg00) (agg03, edg01) (agg12, edg10) (agg12, edg11) (agg13, edg10) (agg13, edg11) (agg22,
 edg20) (agg22, edg21) (agg23, edg20) (agg23, edg21) (agg32, edg30) (agg32, edg31) (agg33, edg30) (agg33, edg31) (cor11, agg02)
 (cor11, agg12) (cor11, agg22) (cor11, agg32) (cor12, agg02) (cor12, agg12) (cor12, agg22) (cor12, agg32) (cor21, agg03) (cor21
, agg13) (cor21, agg23) (cor21, agg33) (cor22, agg03) (cor22, agg13) (cor22, agg23) (cor22, agg33) (edg00, h0.0.2) (edg00, h0.0
.3) (edg01, h0.1.2) (edg01, h0.1.3) (edg10, h1.0.2) (edg10, h1.0.3) (edg11, h1.1.2) (edg11, h1.1.3) (edg20, h2.0.2) (edg20, h2.
0.3) (edg21, h2.1.2) (edg21, h2.1.3) (edg30, h3.0.2) (edg30, h3.0.3) (edg31, h3.1.2) (edg31, h3.1.3)
*** Configuring hosts
h0.0.2 h0.0.3 h0.1.2 h0.1.3 h1.0.2 h1.0.3 h1.1.2 h1.1.3 h2.0.2 h2.0.3 h2.1.2 h2.1.3 h3.0.2 h3.0.3 h3.1.2 h3.1.3
*** Starting controller
c0
*** Starting 20 switches
agg02 agg03 agg12 agg13 agg22 agg23 agg32 agg33 cor11 cor12 cor21 cor22 edg00 edg01 edg10 edg11 edg20 edg21 edg30 edg31 ...
```

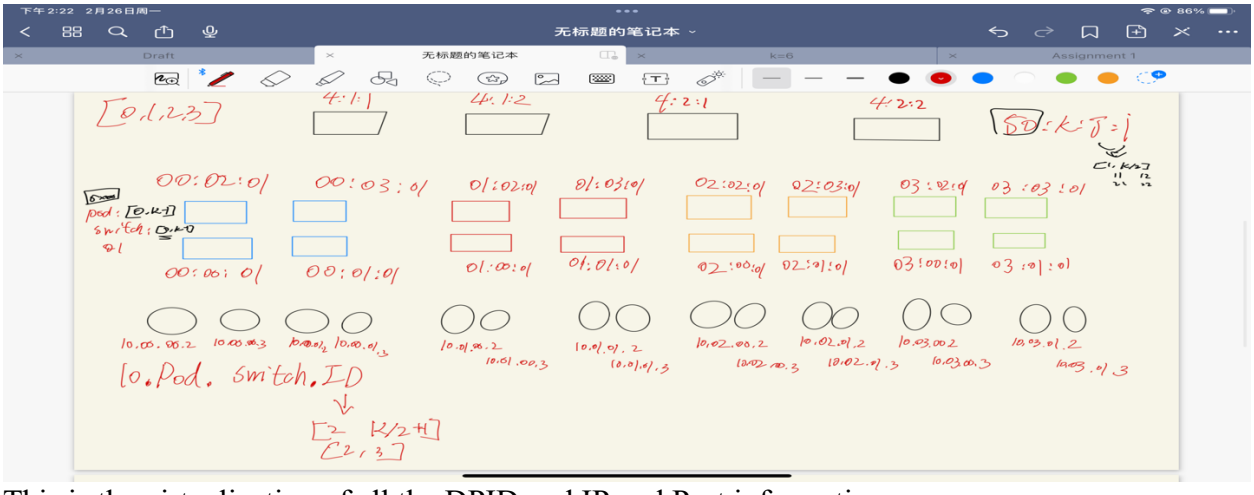
We then run some test to see if the connections for hosts, switches, and ports are good.

```
mininet> links
agg02-eth0<->edg00-eth0 (OK OK)
agg02-eth1<->edg01-eth0 (OK OK)
agg03-eth0<->edg00-eth1 (OK OK)
agg03-eth1<->edg01-eth1 (OK OK)
agg12-eth0<->edg10-eth0 (OK OK)
agg12-eth1<->edg11-eth0 (OK OK)
agg13-eth0<->edg10-eth1 (OK OK)
agg13-eth1<->edg11-eth1 (OK OK)
agg22-eth0<->edg20-eth0 (OK OK)
agg22-eth1<->edg21-eth0 (OK OK)
agg23-eth0<->edg20-eth1 (OK OK)
agg23-eth1<->edg21-eth1 (OK OK)
agg32-eth0<->edg30-eth0 (OK OK)
agg32-eth1<->edg31-eth0 (OK OK)
agg33-eth0<->edg30-eth1 (OK OK)
agg33-eth1<->edg31-eth1 (OK OK)
cor11-eth0<->agg02-eth2 (OK OK)
cor11-eth1<->agg12-eth2 (OK OK)
cor11-eth2<->agg22-eth2 (OK OK)
cor11-eth3<->agg32-eth2 (OK OK)
cor12-eth0<->agg02-eth3 (OK OK)
cor12-eth1<->agg12-eth3 (OK OK)
cor12-eth2<->agg22-eth3 (OK OK)
cor12-eth3<->agg32-eth3 (OK OK)
cor21-eth0<->agg03-eth2 (OK OK)
cor21-eth1<->agg13-eth2 (OK OK)
cor21-eth2<->agg23-eth2 (OK OK)
cor21-eth3<->agg33-eth2 (OK OK)
cor22-eth0<->agg03-eth3 (OK OK)
cor22-eth1<->agg13-eth3 (OK OK)
cor22-eth2<->agg23-eth3 (OK OK)
cor22-eth3<->agg33-eth3 (OK OK)
edg00-eth2<->h0.0.2-eth2 (OK OK)
edg00-eth3<->h0.0.3-eth3 (OK OK)
edg01-eth2<->h0.1.2-eth2 (OK OK)
edg01-eth3<->h0.1.3-eth3 (OK OK)
edg10-eth2<->h1.0.2-eth2 (OK OK)
edg10-eth3<->h1.0.3-eth3 (OK OK)
edg11-eth2<->h1.1.2-eth2 (OK OK)
edg11-eth3<->h1.1.3-eth3 (OK OK)
edg20-eth2<->h2.0.2-eth2 (OK OK)
edg20-eth3<->h2.0.3-eth3 (OK OK)
edg21-eth2<->h2.1.2-eth2 (OK OK)
edg21-eth3<->h2.1.3-eth3 (OK OK)
edg30-eth2<->h3.0.2-eth2 (OK OK)
edg30-eth3<->h3.0.3-eth3 (OK OK)
edg31-eth2<->h3.1.2-eth2 (OK OK)
edg31-eth3<->h3.1.3-eth3 (OK OK)
mininet>
```

There is total 48 links.

```
mininet> dump
<Host h0.0.2: h0.0.2-eth2:10.0.0.0 pid=58452>
<Host h0.0.3: h0.0.3-eth3:10.0.0.1 pid=58454>
<Host h0.1.2: h0.1.2-eth2:10.0.1.0 pid=58456>
<Host h0.1.3: h0.1.3-eth3:10.0.1.1 pid=58458>
<Host h1.0.2: h1.0.2-eth2:10.1.0.0 pid=58460>
<Host h1.0.3: h1.0.3-eth3:10.1.0.1 pid=58462>
<Host h1.1.2: h1.1.2-eth2:10.1.1.0 pid=58464>
<Host h1.1.3: h1.1.3-eth3:10.1.1.1 pid=58466>
<Host h2.0.2: h2.0.2-eth2:10.2.0.0 pid=58468>
<Host h2.0.3: h2.0.3-eth3:10.2.0.1 pid=58470>
<Host h2.1.2: h2.1.2-eth2:10.2.1.0 pid=58472>
<Host h2.1.3: h2.1.3-eth3:10.2.1.1 pid=58474>
<Host h3.0.2: h3.0.2-eth2:10.3.0.0 pid=58476>
<Host h3.0.3: h3.0.3-eth3:10.3.0.1 pid=58478>
<Host h3.1.2: h3.1.2-eth2:10.3.1.0 pid=58480>
<Host h3.1.3: h3.1.3-eth3:10.3.1.1 pid=58482>
<OVSSwitch agg02: agg02-eth0:None,agg02-eth1:None,agg02-eth2:None,agg02-eth3:None pid=58487>
<OVSSwitch agg03: agg03-eth0:None,agg03-eth1:None,agg03-eth2:None,agg03-eth3:None pid=58490>
<OVSSwitch agg12: agg12-eth0:None,agg12-eth1:None,agg12-eth2:None,agg12-eth3:None pid=58493>
<OVSSwitch agg13: agg13-eth0:None,agg13-eth1:None,agg13-eth2:None,agg13-eth3:None pid=58496>
<OVSSwitch agg22: agg22-eth0:None,agg22-eth1:None,agg22-eth2:None,agg22-eth3:None pid=58499>
<OVSSwitch agg23: agg23-eth0:None,agg23-eth1:None,agg23-eth2:None,agg23-eth3:None pid=58502>
<OVSSwitch agg32: agg32-eth0:None,agg32-eth1:None,agg32-eth2:None,agg32-eth3:None pid=58505>
<OVSSwitch agg33: agg33-eth0:None,agg33-eth1:None,agg33-eth2:None,agg33-eth3:None pid=58508>
<OVSSwitch cor11: cor11-eth0:None,cor11-eth1:None,cor11-eth2:None,cor11-eth3:None pid=58511>
<OVSSwitch cor12: cor12-eth0:None,cor12-eth1:None,cor12-eth2:None,cor12-eth3:None pid=58514>
<OVSSwitch cor21: cor21-eth0:None,cor21-eth1:None,cor21-eth2:None,cor21-eth3:None pid=58517>
<OVSSwitch cor22: cor22-eth0:None,cor22-eth1:None,cor22-eth2:None,cor22-eth3:None pid=58520>
<OVSSwitch edg00: edg00-eth0:None,edg00-eth1:None,edg00-eth2:None,edg00-eth3:None pid=58523>
<OVSSwitch edg01: edg01-eth0:None,edg01-eth1:None,edg01-eth2:None,edg01-eth3:None pid=58526>
<OVSSwitch edg10: edg10-eth0:None,edg10-eth1:None,edg10-eth2:None,edg10-eth3:None pid=58529>
<OVSSwitch edg11: edg11-eth0:None,edg11-eth1:None,edg11-eth2:None,edg11-eth3:None pid=58532>
<OVSSwitch edg20: edg20-eth0:None,edg20-eth1:None,edg20-eth2:None,edg20-eth3:None pid=58535>
<OVSSwitch edg21: edg21-eth0:None,edg21-eth1:None,edg21-eth2:None,edg21-eth3:None pid=58538>
<OVSSwitch edg30: edg30-eth0:None,edg30-eth1:None,edg30-eth2:None,edg30-eth3:None pid=58541>
<OVSSwitch edg31: edg31-eth0:None,edg31-eth1:None,edg31-eth2:None,edg31-eth3:None pid=58544>
<Controller c0: 127.0.0.1:6653 pid=58445>
```

This is the result of “dump”, Everything meets the requirements from assignment1.
The Controller C0’s Ip is 127.0.0.1, port is 6653. We will come back to this part later.




This is the virtualization of all the DPID and IP and Port information.

Q1. What is the output for “pingall” command. (Please add a screenshot here.) (5 points)

```
*** Ping: testing ping reachability
h0.0.2 -> h0.0.3 X X X X X X X X X X X X X
h0.0.3 -> h0.0.2 X X X X X X X X X X X X X
h0.1.2 -> X X h0.1.3 X X X X X X X X X X X
h0.1.3 -> X X h0.1.2 X X X X X X X X X X X
h1.0.2 -> X X X X h1.0.3 X X X X X X X X X
h1.0.3 -> X X X X h1.0.2 X X X X X X X X X
h1.1.2 -> X X X X X X X X X X X X X X X
h1.1.3 -> X X X X X X X X X X X X X X X
h2.0.2 -> X X X X X X X X h2.0.3 X X X X X
h2.0.3 -> X X X X X X X X h2.0.2 X X X X X
h2.1.2 -> X X X X X X X X X X X X X X X
h2.1.3 -> X X X X X X X X X X X X X X X
h3.0.2 -> X X X X X X X X X X X X h3.0.3 X X
h3.0.3 -> X X X X X X X X X X X X h3.0.2 X X
h3.1.2 -> X X X X X X X X X X X X X X X
h3.1.3 -> X X X X X X X X X X X X X X X
*** Results: 95% dropped (10/240 received)
```

```
*** Ping: testing ping reachability
h0.0.2 -> h0.0.3 X X X X X X X X X X X X X
h0.0.3 -> h0.0.2 X X X X X X X X X X X X X
h0.1.2 -> X X h0.1.3 X X X X X X X X X X X
h0.1.3 -> X X h0.1.2 X X X X X X X X X X X
h1.0.2 -> X X X X h1.0.3 X X X X X X X X X
h1.0.3 -> X X X X h1.0.2 X X X X X X X X X
h1.1.2 -> X X X X X X X X X X X X X X X
h1.1.3 -> X X X X X X X X X X X X X X X
h2.0.2 -> X X X X X X X X h2.0.3 X X X X X
h2.0.3 -> X X X X X X X X h2.0.2 X X X X X
h2.1.2 -> X X X X X X X X X X X X X X X
h2.1.3 -> X X X X X X X X X X X X X X X
h3.0.2 -> X X X X X X X X X X X X h3.0.3 X X
h3.0.3 -> X X X X X X X X X X X X h3.0.2 X X
h3.1.2 -> X X X X X X X X X X X X X X h3.1.3
h3.1.3 -> X X X X X X X X X X X X X X X
*** Results: 95% dropped (11/240 received)
```

```
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h0.0.2 -> h0.0.3 X X X X X X X X X X X X X
h0.0.3 -> h0.0.2 X X X X X X X X X X X X X
h0.1.2 -> X X h0.1.3 X X X X X X X X X X X
h0.1.3 -> X X h0.1.2 X X X X X X X X X X X
h1.0.2 -> X X X X h1.0.3 X X X X X X X X X
h1.0.3 -> X X X X h1.0.2 X X X X X X X X X
h1.1.2 -> X X X X X X h1.1.3 X X X X X X X
h1.1.3 -> X X X X X X X X X X X X X X X
h2.0.2 -> X X X X X X X X h2.0.3 X X X X X
h2.0.3 -> X X X X X X X X h2.0.2 X X X X X
h2.1.2 -> X X X X X X X X X X X X X X X
h2.1.3 -> X X X X X X X X X X X X X X X
h3.0.2 -> X X X X X X X X X X X X h3.0.3 X X
h3.0.3 -> X X X X X X X X X X X X h3.0.2 X X
h3.1.2 -> X X X X X X X X X X X X X X h3.1.3
h3.1.3 -> X X X X X X X X X X X X X X X
*** Results: 95% dropped (12/240 received)
```

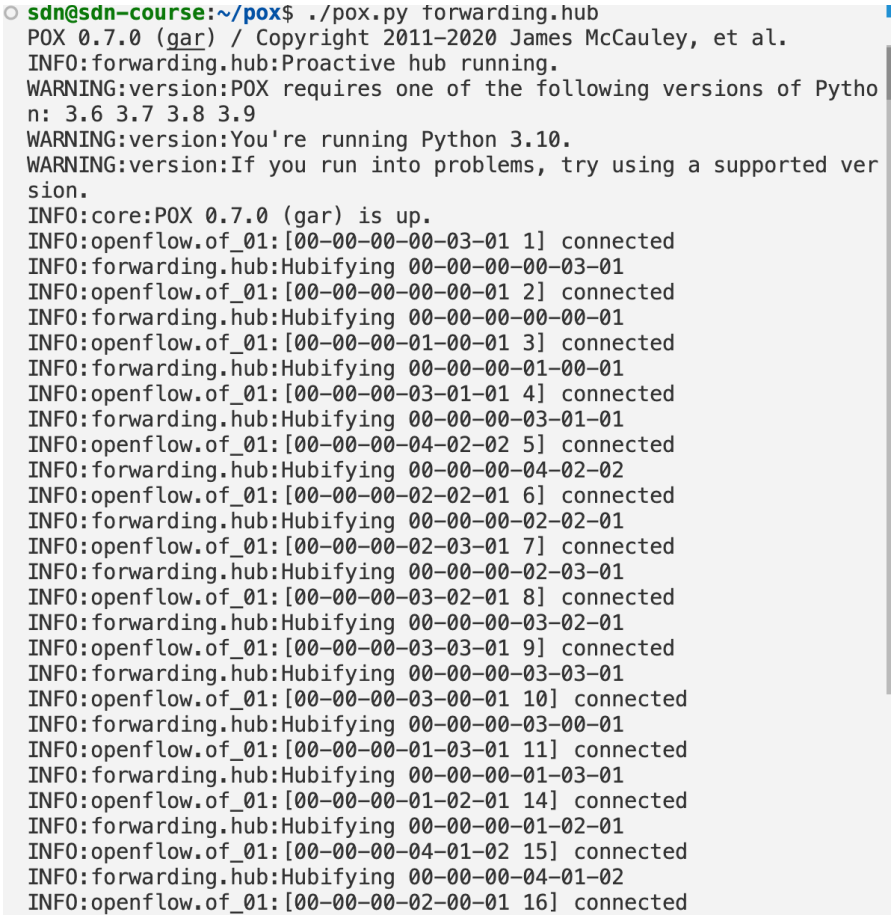
Three screen shots show three different results. Each with a different number of pings received (10 11 12).

Q2. If the result to the “pingall” command is different, (HINT: It should be!) why is that?

The Result is different! I will guide you through my thinking process.

I first thought Open vSwitch would help me to auto fix the problem.

But it did not. Then I thought it might be the problem of Controller, I then give my Fat Tree a ip=127.0.0.1, port=6633(default for POX controller) and connect to POX.




After POX forwarding.hub connected, it shows some DPID. I thought forwarding.hub would regulate the rules for the switches forwarding. The result even worse than before, and I will cover why later.

```
*** Ping: testing ping reachability
h0.0.2 -> X X X X X X X X X X X X X X
h0.0.3 -> X X X X X X X X X X X X X X
h0.1.2 -> X X h0.1.3 X X X X X X X X X X X
h0.1.3 -> X X X X X X X X X X
```

```
INFO:openflow.of_01:[00-00-00-00-02-01 19] connected
INFO:forwarding.hub:Hubifying 00-00-00-00-02-01
INFO:openflow.of_01:[00-00-00-02-01-01 20] connected
INFO:forwarding.hub:Hubifying 00-00-00-02-01-01
INFO:openflow.of_01:[00-00-00-02-01-01 20] closed
INFO:openflow.of_01:[00-00-00-02-01-01 21] connected
INFO:forwarding.hub:Hubifying 00-00-00-02-01-01
INFO:openflow.of_01:[00-00-00-04-02-01 18] closed
INFO:openflow.of_01:[00-00-00-00-02-01 19] closed
INFO:openflow.of_01:[00-00-00-04-02-01 24] connected
INFO:forwarding.hub:Hubifying 00-00-00-04-02-01
INFO:openflow.of_01:[00-00-00-00-02-01 25] connected
INFO:forwarding.hub:Hubifying 00-00-00-00-02-01
INFO:openflow.of_01:[00-00-00-02-00-01 16] closed
INFO:openflow.of_01:[00-00-00-02-00-01 26] connected
INFO:forwarding.hub:Hubifying 00-00-00-02-00-01
INFO:openflow.of_01:[00-00-00-00-01-01 13] closed
INFO:openflow.of_01:[00-00-00-00-01-01 27] connected
INFO:forwarding.hub:Hubifying 00-00-00-00-01-01
INFO:openflow.of_01:[00-00-00-01-01-01 12] closed
INFO:openflow.of_01:[00-00-00-01-01-01 28] connected
INFO:forwarding.hub:Hubifying 00-00-00-01-01-01
```

I then tried forwarding.l2_learning, because it will check the source MAC address of the packet and records the input port associated with that address. I hoped this would help the host to receive the packet and respond. The result turnout to be better than using forwarding.hub, but still has 95% of drop rate.

```
sdn@sdn-course:~/pox$ ./pox.py forwarding.l2_learning
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
WARNING:version:POX requires one of the following versions of Pytho
n: 3.6 3.7 3.8 3.9
WARNING:version:You're running Python 3.10.
WARNING:version:If you run into problems, try using a supported ver
sion.
INFO:core:POX 0.7.0 (gar) is up.
INFO:openflow.of_01:[00-00-00-02-03-01 17] connected
INFO:openflow.of_01:[00-00-00-01-02-01 4] connected
INFO:openflow.of_01:[00-00-00-01-03-01 1] connected
INFO:openflow.of_01:[00-00-00-01-01-01 2] connected
INFO:openflow.of_01:[00-00-00-00-01-01 3] connected
INFO:openflow.of_01:[00-00-00-04-01-02 5] connected
INFO:openflow.of_01:[00-00-00-02-00-01 6] connected
INFO:openflow.of_01:[00-00-00-04-01-01 7] connected
INFO:openflow.of_01:[00-00-00-04-02-01 8] connected
INFO:openflow.of_01:[00-00-00-00-02-01 9] connected
INFO:openflow.of_01:[00-00-00-02-01-01 10] connected
INFO:openflow.of_01:[00-00-00-00-03-01 11] connected
INFO:openflow.of_01:[00-00-00-00-00-01 12] connected
INFO:openflow.of_01:[00-00-00-01-00-01 13] connected
```

```
mininet> pingall
*** Ping: testing ping reachability
h0.0.2 -> h0.0.3 X X X X X X X X X X X X
h0.0.3 -> h0.0.2 X X X X X X X X X X X X
h0.1.2 -> X X h0.1.3 X X X X X X X X X X
h0.1.3 -> X X h0.1.2 X X X X X X X X X X
h1.0.2 -> X X X h1.0.3 X X X X X X X X X
h1.0.3 -> X X X h1.0.2 X X X X X X X X X
h1.1.2 -> X X X X X X X X X X X X X
h1.1.3 -> X X X X X X X X X X X X X
h2.0.2 -> X X X X X X X X h2.0.3 X X X X X
h2.0.3 -> X X X X X X X X h2.0.2 X X X X X
h2.1.2 -> X X X X X X X X X h2.1.3 X X X X
```

```
INFO:openflow.of_01:[00-00-00-00-03-01 11] connected
INFO:openflow.of_01:[00-00-00-00-00-01 12] connected
INFO:openflow.of_01:[00-00-00-01-00-01 13] connected
INFO:openflow.of_01:[00-00-00-03-01-01 14] connected
INFO:openflow.of_01:[00-00-00-04-02-02 15] connected
INFO:openflow.of_01:[00-00-00-02-02-01 16] connected
INFO:openflow.of_01:[00-00-00-03-02-01 18] connected
INFO:openflow.of_01:[00-00-00-03-03-01 19] connected
INFO:openflow.of_01:[00-00-00-03-00-01 20] connected

WARNING:forwarding.l2_learning:Same port for packet from 00:00:00:0
0:00:08 -> 00:00:00:00:00:07 on 00-00-00-01-03-01.1.  Drop.
WARNING:forwarding.l2_learning:Same port for packet from 00:00:00:0
0:00:08 -> 00:00:00:00:00:07 on 00-00-00-01-03-01.1.  Drop.
WARNING:forwarding.l2_learning:Same port for packet from 00:00:00:0
0:00:08 -> 00:00:00:00:00:07 on 00-00-00-03-03-01.3.  Drop.
WARNING:forwarding.l2_learning:Same port for packet from 00:00:00:0
0:00:08 -> 00:00:00:00:00:07 on 00-00-00-03-03-01.2.  Drop.
WARNING:forwarding.l2_learning:Same port for packet from 00:00:00:0
0:00:08 -> 00:00:00:00:00:07 on 00-00-00-03-03-01.2.  Drop.
```

After that, I did bunch of study and research. I found out that the problem was due to broadcast storm.

Based on the structure of Fat Tree, there are so many loops inside. The loops were caused by the number of connections. When a loop happens, the packets will repeatedly transmit within the broadcast storm.
To solve this, I found out there are 2 ways, but I don't know how to implement those. First way is to enable broadcast storm suppression on switches, I guess only the higher-level switches are supported. The second way is to apply the STP spanning tree protocol or any other loop-breaking techniques.
That is probably the reason why forwarding.hub made the result even worse, only 1 packet received instead of 11 on average (out of 240).
Also, I found out that POX is way to light weighted, the ./pox.py openflow.spanning_tree in the POX was that useful. Floodlight can do much better job without any implementation. Ryu is also a potential fix to this broadcast storm.