Lab3 Report: SDN Open Virtual Switches

** Please **fill in the report** and submit the **pdf** to NYUClasses*

Name: Shengwei Huang ID: Sh6203 Date: 11/11/2020

1. Objectives

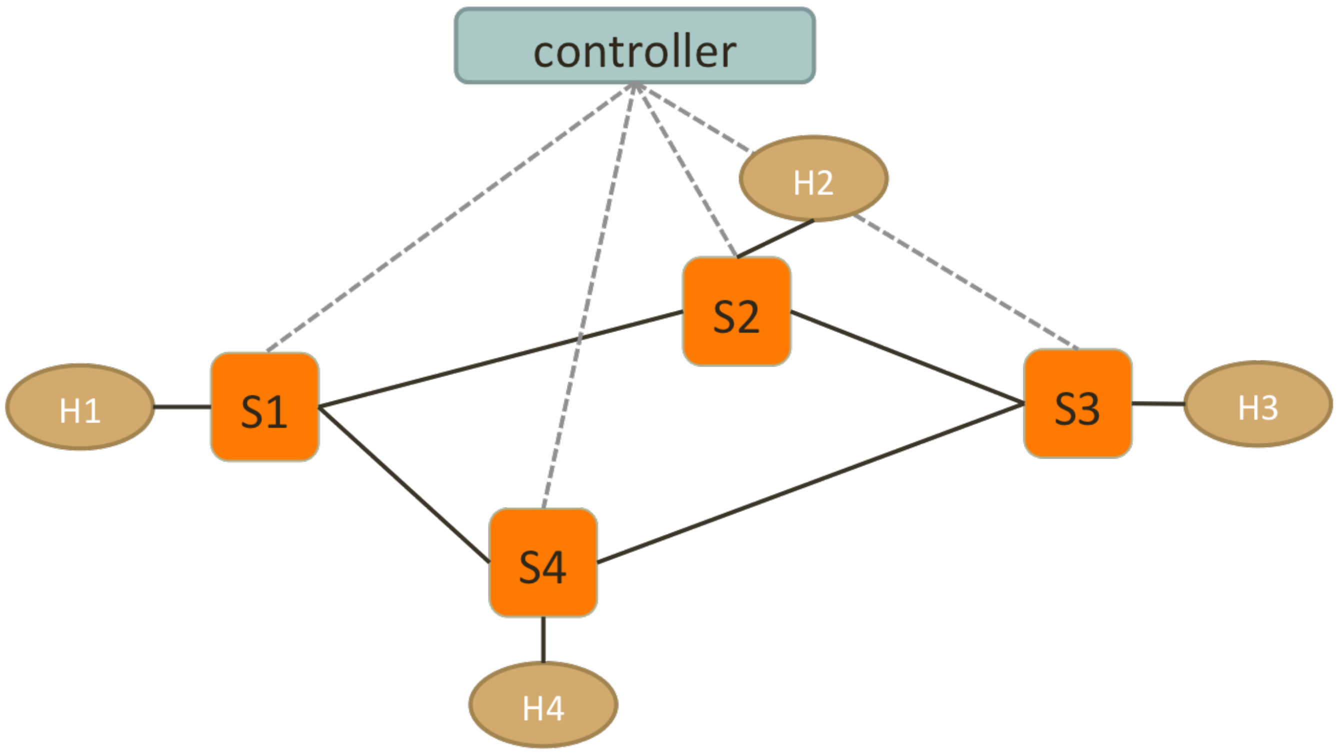
- Understand SDN and get familiar with controllers.

2. References

- https://github.com/faucetsdn/ryu/blob/master/ryu/app/simple_switch_13.py
- https://ryu.readthedocs.io/en/latest/ofproto_v1_3_ref.html
- Slides

3. Experiments

1. Use Mininet to create the following topology: (4 Hosts, 4 OVSes) with a remote controller
2. Use RYU to implement the controller (you can use other controller such as BEACON, POX, etc...)



3. Test Connectivity using ping. (Hint: take care of ARP packets in the controller and install proper rules for them.)
4. Enforce *these policies*:
   - **Everything follows shortest path**
   - **When there are two shortest paths with equal costs available**
     - ICMP and TCP packets take the clockwise path
       - e.g. S1-S2-S3, S2-S3-S4
     - UDP packets take the counterclockwise path
       - e.g. S1-S4-S3, S2-S1-S4
     - H2 and H4 cannot send HTTP traffic (TCP with dst_port:80)
       - New connections are dropped with a TCP RST sent back to **H2 or H4**
       - To be more specific, when the first TCP packet (SYN) arrives **S2 or S4**, forwarded it to controller, controller then create a RST packet and send it back to the host.
     - H1 and H4 cannot send UDP traffic
       - simply drop packets at switches

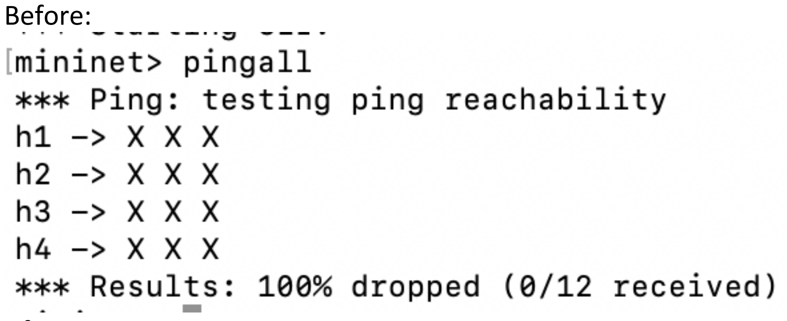
Important! Handle the flow rules in Packet-In and let the controller handles the rules dynamically.

If you use static rules for those policies or handle them in SwitchFeatureHandler, your lab score will be removed.

4. Reports

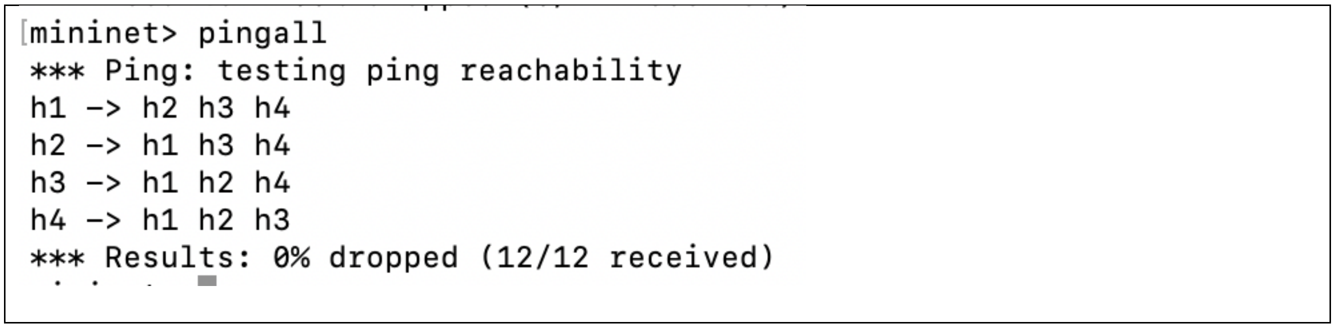
(a) Screenshots of your mininet with "pingall", **before** and **after starting the controller**.

Before:



```
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)
```

After:

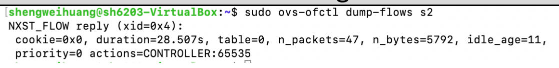
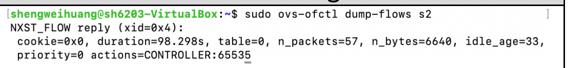
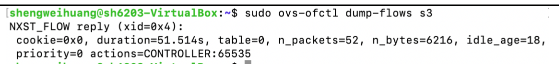
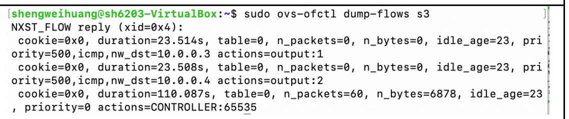


```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

(b) How do you generate different traffic? Which tools do you use to generate: ICMP, TCP, UDP and HTTP traffic?

Icmp: h1 ping h2
Tcp: h1 hping3 h2   receiver: iperf -s -p sender: iperf -c host -p
http: h1 hping3 h2 -p 80 receiver: iperf -s -p 80   Sender: iperf -c host -p 80
udp: h1 hping3 h2 -2   Receiver: iperf -s -u -p Sender: iperf -c host -u -p

(c) Generate ICMP flows from **H4 to H3**, and take **screenshots** of the flow table on **S2** and S3 before and after the flow is generated to show that your flow follow the right path. (ovs-ofctl dump-flows)

	Before ICMP flow is generated	After ICMP flow is generated
S2		
S3		

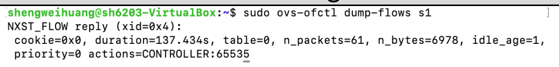
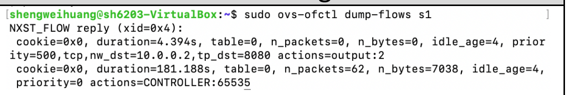
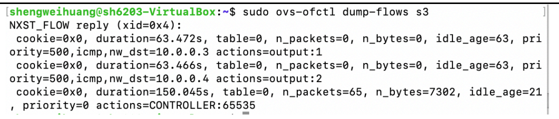
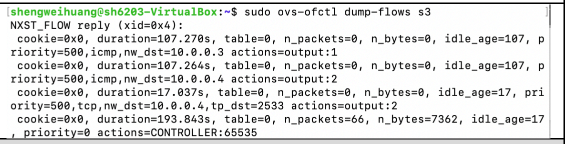
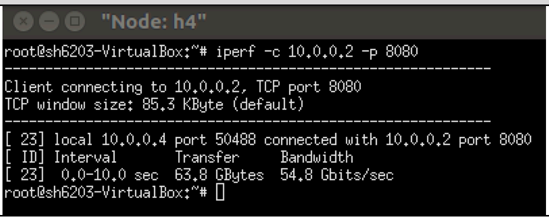
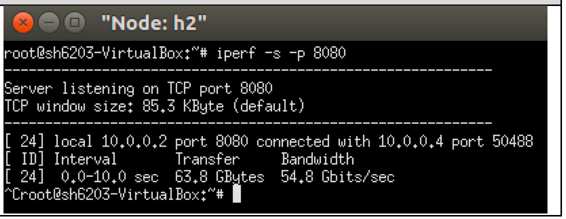
```
shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s2
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=28.507s, table=0, n_packets=47, n_bytes=5792, idle_age=11,
 priority=0 actions=CONTROLLER:65535
```

```
shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s2
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=98.298s, table=0, n_packets=57, n_bytes=6640, idle_age=33,
 priority=0 actions=CONTROLLER:65535
```

```
shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=51.514s, table=0, n_packets=52, n_bytes=6216, idle_age=18,
 priority=0 actions=CONTROLLER:65535
```

```
shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=23.514s, table=0, n_packets=0, n_bytes=0, idle_age=23, pri
ority=500,icmp,nw_dst=10.0.0.3 actions=output:1
 cookie=0x0, duration=23.508s, table=0, n_packets=0, n_bytes=0, idle_age=23, pri
ority=500,icmp,nw_dst=10.0.0.4 actions=output:2
 cookie=0x0, duration=110.087s, table=0, n_packets=60, n_bytes=6878, idle_age=23
, priority=0 actions=CONTROLLER:65535
```

(d) Generate TCP flows (dst_port: 8080) from **H4 to H2**, and take **screenshots** of the flow table on S1 and S3 before and after the flow is generated. (ovs-ofctl dump-flows) Also, the screenshot of your Mininet or host that generates/receives the TCP traffic.

	Before TCP flow is generated	After TCP flow is generated
S1		
S3		
	Generates TCP traffic	Receives TCP traffic
Mininet or hosts		

```
shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=137.434s, table=0, n_packets=61, n_bytes=6978, idle_age=1,
 priority=0 actions=CONTROLLER:65535
```

```
shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=4.394s, table=0, n_packets=0, n_bytes=0, idle_age=4, prior
ity=500,tcp,nw_dst=10.0.0.2,tp_dst=8080 actions=output:2
 cookie=0x0, duration=181.188s, table=0, n_packets=62, n_bytes=7038, idle_age=4,
 priority=0 actions=CONTROLLER:65535
```

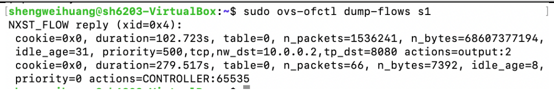
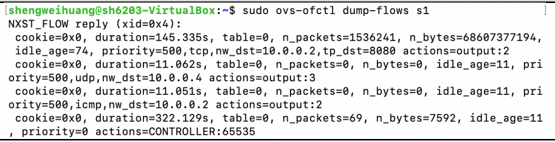
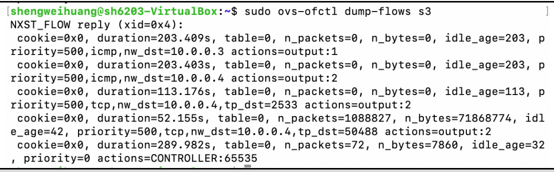
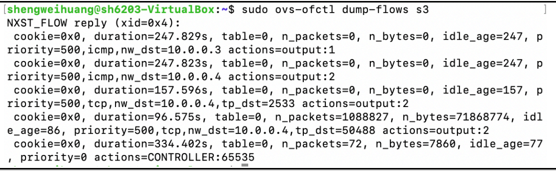
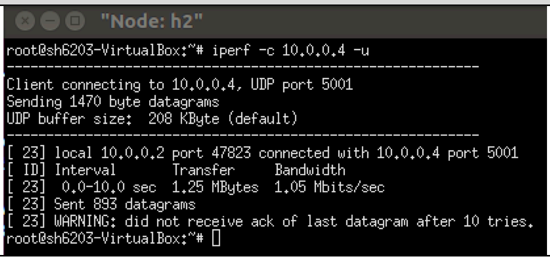
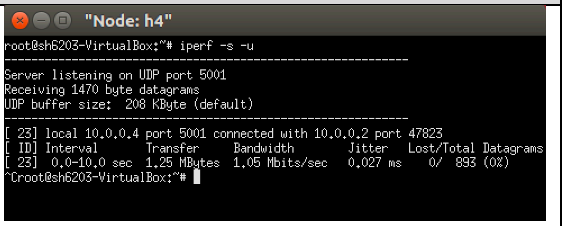
```
shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=63.472s, table=0, n_packets=0, n_bytes=0, idle_age=63, pri
ority=500,icmp,nw_dst=10.0.0.3 actions=output:1
 cookie=0x0, duration=63.466s, table=0, n_packets=0, n_bytes=0, idle_age=63, pri
ority=500,icmp,nw_dst=10.0.0.4 actions=output:2
 cookie=0x0, duration=150.045s, table=0, n_packets=65, n_bytes=7302, idle_age=21
, priority=0 actions=CONTROLLER:65535
```

```
shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=107.270s, table=0, n_packets=0, n_bytes=0, idle_age=107, p
riority=500,icmp,nw_dst=10.0.0.3 actions=output:1
 cookie=0x0, duration=107.264s, table=0, n_packets=0, n_bytes=0, idle_age=107, p
riority=500,icmp,nw_dst=10.0.0.4 actions=output:2
 cookie=0x0, duration=17.037s, table=0, n_packets=0, n_bytes=0, idle_age=17, pri
ority=500,tcp,nw_dst=10.0.0.4,tp_dst=2533 actions=output:2
 cookie=0x0, duration=193.843s, table=0, n_packets=66, n_bytes=7362, idle_age=17
, priority=0 actions=CONTROLLER:65535
```

```
root@sh6203-VirtualBox:~# iperf -c 10.0.0.2 -p 8080
------------------------------------------------------------
Client connecting to 10.0.0.2, TCP port 8080
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 23] local 10.0.0.4 port 50488 connected with 10.0.0.2 port 8080
[ ID] Interval       Transfer     Bandwidth
[ 23]  0.0-10.0 sec  63.8 GBytes  54.8 Gbits/sec
root@sh6203-VirtualBox:~# 
```

```
root@sh6203-VirtualBox:~# iperf -s -p 8080
------------------------------------------------------------
Server listening on TCP port 8080
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 24] local 10.0.0.2 port 8080 connected with 10.0.0.4 port 50488
[ ID] Interval       Transfer     Bandwidth
[ 24]  0.0-10.0 sec  63.8 GBytes  54.8 Gbits/sec
^Croot@sh6203-VirtualBox:~# 
```

(e) Generate UDP flows from H2 to H4, and take **screenshots** of the flow table on S1 and S3 before and after the flow is generated. (ovs-ofctl dump-flows) Also, the screenshot of your Mininet or host that generates/receives the UDP traffic.

	Before UDP flow is generated	After UDP flow is generated
S1		
S3		
	Generates UDP traffic	**Receives UDP traffic**
Mininet or hosts		

S1, before:

```
[shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s1                ]
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=102.723s, table=0, n_packets=1536241, n_bytes=68607377194,
 idle_age=31, priority=500,tcp,nw_dst=10.0.0.2,tp_dst=8080 actions=output:2
 cookie=0x0, duration=279.517s, table=0, n_packets=66, n_bytes=7392, idle_age=8,
 priority=0 actions=CONTROLLER:65535
```

S1, after:

```
[shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s1                ]
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=145.335s, table=0, n_packets=1536241, n_bytes=68607377194,
 idle_age=74, priority=500,tcp,nw_dst=10.0.0.2,tp_dst=8080 actions=output:2
 cookie=0x0, duration=11.062s, table=0, n_packets=0, n_bytes=0, idle_age=11, pri
ority=500,udp,nw_dst=10.0.0.4 actions=output:3
 cookie=0x0, duration=11.051s, table=0, n_packets=0, n_bytes=0, idle_age=11, pri
ority=500,icmp,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=322.129s, table=0, n_packets=69, n_bytes=7592, idle_age=11
, priority=0 actions=CONTROLLER:65535
```

S3, before:

```
[shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s3                ]
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=203.409s, table=0, n_packets=0, n_bytes=0, idle_age=203, p
riority=500,icmp,nw_dst=10.0.0.3 actions=output:1
 cookie=0x0, duration=203.403s, table=0, n_packets=0, n_bytes=0, idle_age=203, p
riority=500,icmp,nw_dst=10.0.0.4 actions=output:2
 cookie=0x0, duration=113.176s, table=0, n_packets=0, n_bytes=0, idle_age=113, p
riority=500,tcp,nw_dst=10.0.0.4,tp_dst=2533 actions=output:2
 cookie=0x0, duration=52.155s, table=0, n_packets=1088827, n_bytes=71868774, idl
e_age=42, priority=500,tcp,nw_dst=10.0.0.4,tp_dst=50488 actions=output:2
 cookie=0x0, duration=289.982s, table=0, n_packets=72, n_bytes=7860, idle_age=32
, priority=0 actions=CONTROLLER:65535
```

S3, after:

```
[shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s3                ]
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=247.829s, table=0, n_packets=0, n_bytes=0, idle_age=247, p
riority=500,icmp,nw_dst=10.0.0.3 actions=output:1
 cookie=0x0, duration=247.823s, table=0, n_packets=0, n_bytes=0, idle_age=247, p
riority=500,icmp,nw_dst=10.0.0.4 actions=output:2
 cookie=0x0, duration=157.596s, table=0, n_packets=0, n_bytes=0, idle_age=157, p
riority=500,tcp,nw_dst=10.0.0.4,tp_dst=2533 actions=output:2
 cookie=0x0, duration=96.575s, table=0, n_packets=1088827, n_bytes=71868774, idl
e_age=86, priority=500,tcp,nw_dst=10.0.0.4,tp_dst=50488 actions=output:2
 cookie=0x0, duration=334.402s, table=0, n_packets=72, n_bytes=7860, idle_age=77
, priority=0 actions=CONTROLLER:65535
```

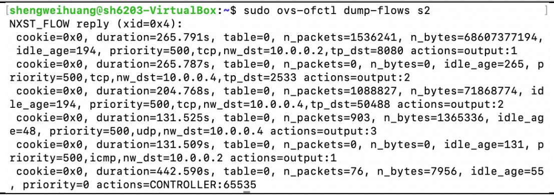
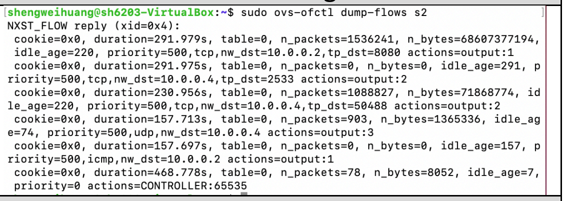
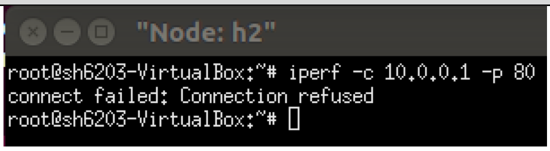
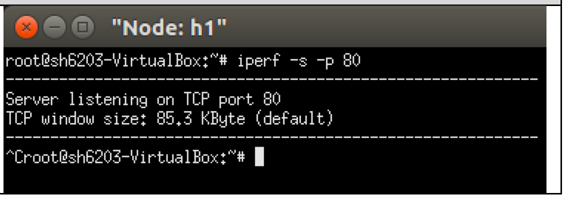
"Node: h2"

```
root@sh6203-VirtualBox:~# iperf -c 10.0.0.4 -u
------------------------------------------------------------
Client connecting to 10.0.0.4, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[ 23] local 10.0.0.2 port 47823 connected with 10.0.0.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 23]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[ 23] Sent 893 datagrams
[ 23] WARNING: did not receive ack of last datagram after 10 tries.
root@sh6203-VirtualBox:~# 
```

"Node: h4"

```
root@sh6203-VirtualBox:~# iperf -s -u
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[ 23] local 10.0.0.4 port 5001 connected with 10.0.0.2 port 47823
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[ 23]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.027 ms    0/  893 (0%)
^Croot@sh6203-VirtualBox:~# 
```

(f) Generate HTTP traffic from H2 to H1, and take **screenshots** of the flow table on S2 before and after the flow is generated. (ovs-ofctl dump-flows) Also, the screenshot of your Mininet or host that generates/receives the HTTP traffic.

	Before HTTP flow is generated	After HTTP flow is generated
S2		
	Generates HTTP traffic	**Receives HTTP traffic**
Mininet or hosts		

S2, before:

```
[shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s2                ]
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=265.791s, table=0, n_packets=1536241, n_bytes=68607377194,
 idle_age=194, priority=500,tcp,nw_dst=10.0.0.2,tp_dst=8080 actions=output:1
 cookie=0x0, duration=265.787s, table=0, n_packets=0, n_bytes=0, idle_age=265, p
riority=500,tcp,nw_dst=10.0.0.4,tp_dst=2533 actions=output:2
 cookie=0x0, duration=204.768s, table=0, n_packets=1088827, n_bytes=71868774, id
le_age=194, priority=500,tcp,nw_dst=10.0.0.4,tp_dst=50488 actions=output:2
 cookie=0x0, duration=131.525s, table=0, n_packets=903, n_bytes=1365336, idle_ag
e=48, priority=500,udp,nw_dst=10.0.0.4 actions=output:3
 cookie=0x0, duration=131.509s, table=0, n_packets=0, n_bytes=0, idle_age=131, p
riority=500,icmp,nw_dst=10.0.0.2 actions=output:1
 cookie=0x0, duration=442.590s, table=0, n_packets=76, n_bytes=7956, idle_age=55
, priority=0 actions=CONTROLLER:65535
```

S2, after:

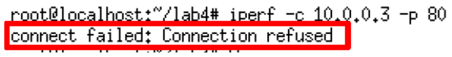
```
[shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s2                ]
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=291.979s, table=0, n_packets=1536241, n_bytes=68607377194,
 idle_age=220, priority=500,tcp,nw_dst=10.0.0.2,tp_dst=8080 actions=output:1
 cookie=0x0, duration=291.975s, table=0, n_packets=0, n_bytes=0, idle_age=291, p
riority=500,tcp,nw_dst=10.0.0.4,tp_dst=2533 actions=output:2
 cookie=0x0, duration=230.956s, table=0, n_packets=1088827, n_bytes=71868774, id
le_age=220, priority=500,tcp,nw_dst=10.0.0.4,tp_dst=50488 actions=output:2
 cookie=0x0, duration=157.713s, table=0, n_packets=903, n_bytes=1365336, idle_ag
e=74, priority=500,udp,nw_dst=10.0.0.4 actions=output:3
 cookie=0x0, duration=157.697s, table=0, n_packets=0, n_bytes=0, idle_age=157, p
riority=500,icmp,nw_dst=10.0.0.2 actions=output:1
 cookie=0x0, duration=468.778s, table=0, n_packets=78, n_bytes=8052, idle_age=7,
 priority=0 actions=CONTROLLER:65535
```

"Node: h2"

```
root@sh6203-VirtualBox:~# iperf -c 10.0.0.1 -p 80
connect failed: Connection refused
root@sh6203-VirtualBox:~# 
```

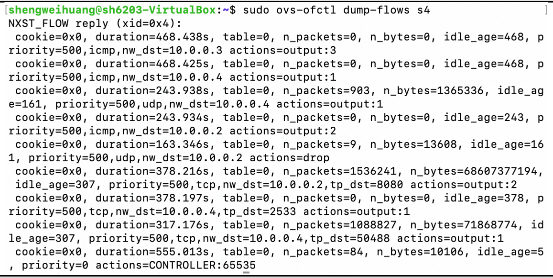
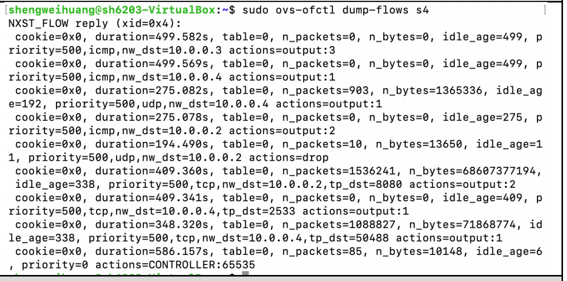
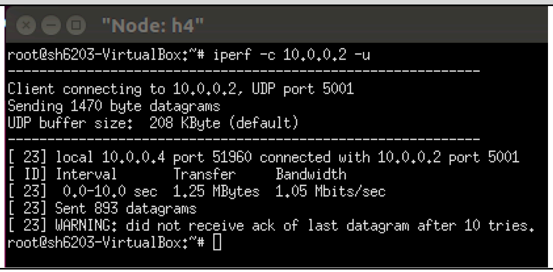
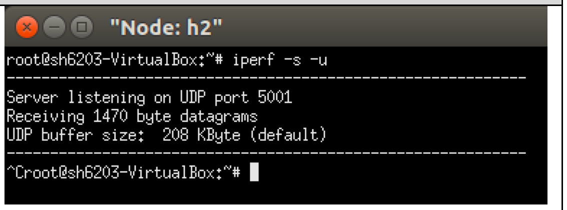
"Node: h1"

```
root@sh6203-VirtualBox:~# iperf -s -p 80
------------------------------------------------------------
Server listening on TCP port 80
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
^Croot@sh6203-VirtualBox:~# 
```

Note: "**Connection refused**" means the RST packets is successfully sent back to S2. Otherwise, you need to check if your RST packets is correct. e.g., 

```
root@localhost:~/lab4# iperf -c 10.0.0.3 -p 80
connect failed: Connection refused
```

(g) Generate UDP traffic from H4 to H2, and take **screenshots** of the flow table on S4 before and after the flow is generated. (ovs-ofctl dump-flows) Also, the screenshot of your Mininet or host that generates/receives the UDP traffic.

	Before UDP flow is generated	After UDP flow is generated
S4		
	Generates UDP traffic	**Receives UDP traffic**
Mininet or hosts		

```
shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s4
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=468.438s, table=0, n_packets=0, n_bytes=0, idle_age=468, p
riority=500,icmp,nw_dst=10.0.0.3 actions=output:3
 cookie=0x0, duration=468.425s, table=0, n_packets=0, n_bytes=0, idle_age=468, p
riority=500,icmp,nw_dst=10.0.0.4 actions=output:1
 cookie=0x0, duration=243.938s, table=0, n_packets=903, n_bytes=1365336, idle_ag
e=161, priority=500,udp,nw_dst=10.0.0.4 actions=output:1
 cookie=0x0, duration=243.934s, table=0, n_packets=0, n_bytes=0, idle_age=243, p
riority=500,icmp,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=163.346s, table=0, n_packets=9, n_bytes=13608, idle_age=16
1, priority=500,udp,nw_dst=10.0.0.2 actions=drop
 cookie=0x0, duration=378.216s, table=0, n_packets=1536241, n_bytes=68607377194,
 idle_age=307, priority=500,tcp,nw_dst=10.0.0.2,tp_dst=8080 actions=output:2
 cookie=0x0, duration=378.197s, table=0, n_packets=0, n_bytes=0, idle_age=378, p
riority=500,tcp,nw_dst=10.0.0.4,tp_dst=2533 actions=output:1
 cookie=0x0, duration=317.176s, table=0, n_packets=1088827, n_bytes=71868774, id
le_age=307, priority=500,tcp,nw_dst=10.0.0.4,tp_dst=50488 actions=output:1
 cookie=0x0, duration=555.013s, table=0, n_packets=84, n_bytes=10106, idle_age=5
, priority=0 actions=CONTROLLER:65535
```

```
shengweihuang@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s4
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=499.582s, table=0, n_packets=0, n_bytes=0, idle_age=499, p
riority=500,icmp,nw_dst=10.0.0.3 actions=output:3
 cookie=0x0, duration=499.569s, table=0, n_packets=0, n_bytes=0, idle_age=499, p
riority=500,icmp,nw_dst=10.0.0.4 actions=output:1
 cookie=0x0, duration=275.082s, table=0, n_packets=903, n_bytes=1365336, idle_ag
e=192, priority=500,udp,nw_dst=10.0.0.4 actions=output:1
 cookie=0x0, duration=275.078s, table=0, n_packets=0, n_bytes=0, idle_age=275, p
riority=500,icmp,nw_dst=10.0.0.2 actions=output:2
 cookie=0x0, duration=194.490s, table=0, n_packets=10, n_bytes=13650, idle_age=1
1, priority=500,udp,nw_dst=10.0.0.2 actions=drop
 cookie=0x0, duration=409.360s, table=0, n_packets=1536241, n_bytes=68607377194,
 idle_age=338, priority=500,tcp,nw_dst=10.0.0.2,tp_dst=8080 actions=output:2
 cookie=0x0, duration=409.341s, table=0, n_packets=0, n_bytes=0, idle_age=409, p
riority=500,tcp,nw_dst=10.0.0.4,tp_dst=2533 actions=output:1
 cookie=0x0, duration=348.320s, table=0, n_packets=1088827, n_bytes=71868774, id
le_age=338, priority=500,tcp,nw_dst=10.0.0.4,tp_dst=50488 actions=output:1
 cookie=0x0, duration=586.157s, table=0, n_packets=85, n_bytes=10148, idle_age=6
, priority=0 actions=CONTROLLER:65535
```

"Node: h4"

```
root@sh6203-VirtualBox:~# iperf -c 10.0.0.2 -u
------------------------------------------------------------
Client connecting to 10.0.0.2, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[ 23] local 10.0.0.4 port 51960 connected with 10.0.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 23]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[ 23] Sent 893 datagrams
[ 23] WARNING: did not receive ack of last datagram after 10 tries.
root@sh6203-VirtualBox:~# 
```

"Node: h2"

```
root@sh6203-VirtualBox:~# iperf -s -u
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
^Croot@sh6203-VirtualBox:~# 
```

(h) Please find what is "Spanning Tree" and "Spanning Tree Protocol"? What's the purpose of the protocol?

Spanning tree is included in tree graph. It has vertices, and all the vertices will have minimum possible number of edges.

Spanning tree protocol is a network protocol which is used to create a loop-free logical topology.

Purpose of spanning tree protocol: create loop-free logical topology in the network to avoid flood.

(i) Is it necessary to implement spanning tree in SDN for packet forwarding? Why?

No, we don't need to implement spanning tree in SDN for packet forwarding. Because SDN is flexible enough, it has 5 tuples which can be used to implement packet forwarding. So we do not need to implement spanning tree in SDN for packet forwarding.

(j) If you want to find spanning tree in SDN, how will you implement and what is the difference between traditional "Spanning Tree Protocol" and the one in SDN?

I will use 5 tuples to implement a spanning tree in SDN. For SDN we know it is flexible enough so we do not need to break down any connection like what spanning tree protocol does. We only need to use SDN controller to control the flow in the network and avoid the loop. So we will have the spanning tree in SDN.

For the spanning Tree Protocol, we need to break down some connection to avoid the loop. However, this way will have some resources lost. In the SDN we do not need to break down any connection so we can avoid the loop.

(k) List three advantages of using OpenVSwitch and SDN controller compared to IP networks. Briefly explain why

1. Compare to the IP network SDN is a centralized network provisioning. It can control all the network according to the centralized controller.
2. Compare to the IP network the SDN has lower operating cost.
3. SDN can be more flexible than IP network. It use a central controller and 5 tuples to implement any

flows in the netwrok. And due to the 5 tuples, it can achieve lots of functions which IP network cannot achieve.

(l) Include the controller's code.

(Upload with your report or attach a sharable link)
https://drive.google.com/file/d/1mIxPhe7Hx4-upEvfqWPb2hsAwCy0yj3M/view?usp=sharing

(m) Include the topology file

(Upload with your report or attach a sharable link)
https://drive.google.com/file/d/1_2mj-2rb5U63AjXp01VgEcvhb2-KC6gx/view?usp=sharing

(n) Challenges you've encountered while doing this experiment, and explain how you manage to solve them. If you do not experience any problem, simply say no problem.

1. I meet problem to understand the each layer in this lab. Then I watch the video on the nyu class and rea the pdf files to understand the code and layer. Then I figure out how to implement each layer and packet and how to add flows on it.
2. I meet some problems on calculate the out port. Firstly, I decide to write all the path out. However, this way is too complex. Then I find the outport can be the same in the clockwise. And outport can also be the same in the counterclockwise. So I just use mathematic way to calculate the outport and the shortest path way out.

We have zero tolerance to forged or fabricated data!! A single piece of forged/fabricated data would bring the total score down to zero.