

# Lab3 Report: SDN Open Virtual Switches

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

Name:           Ziyan Qiu           ID:           zq2048           Date:           11.7.2021          

## 1. Objectives

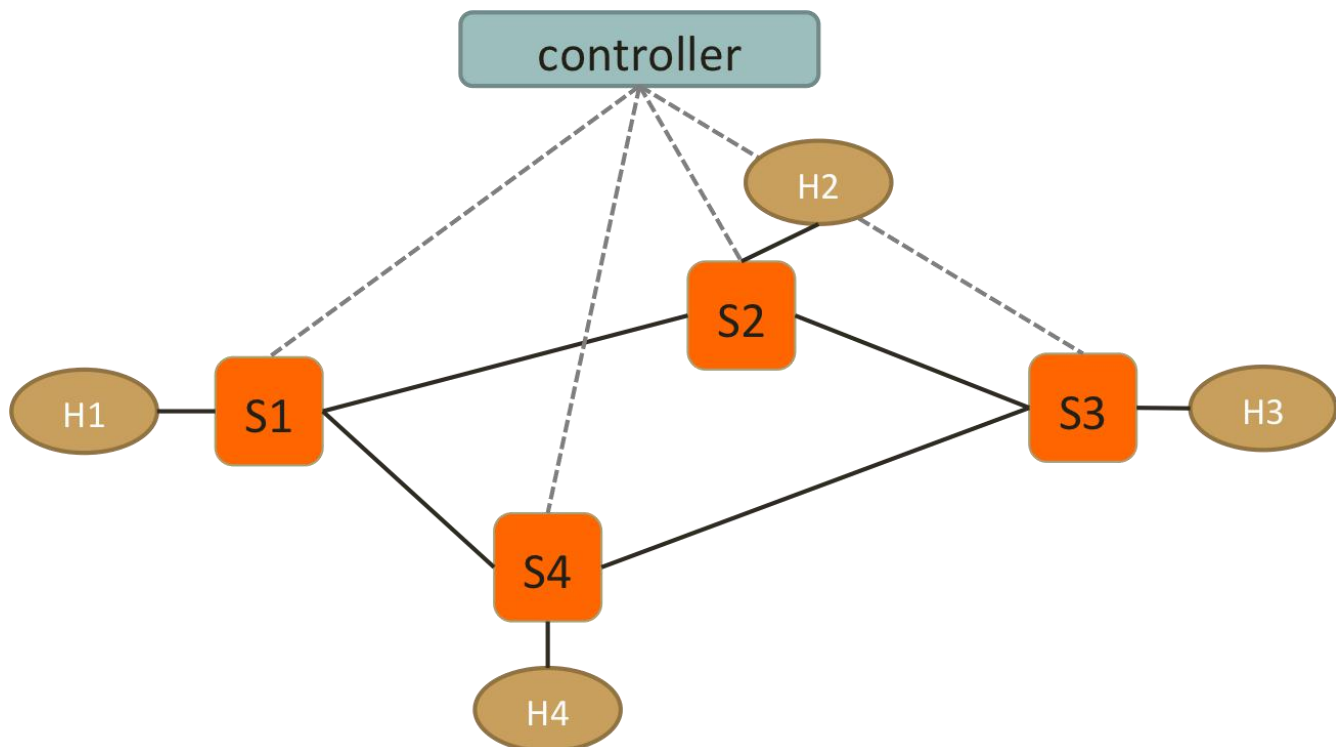
- Understand SDN and get familiar with controllers.

## 2. References

- [https://github.com/faucetsdn/ryu/blob/master/ryu/app/simple\\_switch\\_13.py](https://github.com/faucetsdn/ryu/blob/master/ryu/app/simple_switch_13.py)
- [https://ryu.readthedocs.io/en/latest/ofproto\\_v1\\_3\\_ref.html](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: ping  
TCP: iperf  
UDP: iperf  
HTTP: iperf

- (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	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s2 --protocols=OpenFlow13 OFPST_FLOW reply (OF1.3) (xid=0x2):  cookie=0x0, duration=19.734s, table=0, n_packets=42, n_bytes=2142, priority=655  35,d1,dst=01:80:c2:00:00:0e,d1_type=0x80cc actions=CONTROLLER:65535  cookie=0x0, duration=19.738s, table=0, n_packets=1, n_bytes=51, priority=0 actions=CONTROLLER:65535  root@qzy-VirtualBox:/media#</pre>	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s2 --protocols=OpenFlow13 OFPST_FLOW reply (OF1.3) (xid=0x2):  cookie=0x0, duration=19.734s, table=0, n_packets=189, n_bytes=5180, priority=65  35,d1,dst=01:80:c2:00:00:0e,d1_type=0x80cc actions=CONTROLLER:65535  cookie=0x0, duration=45.739s, table=0, n_packets=13, n_bytes=1367, priority=0 actions=CONTROLLER:65535</pre>
S3	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s3 --protocols=OpenFlow13 OFPST_FLOW reply (OF1.3) (xid=0x2):  cookie=0x0, duration=23.857s, table=0, n_packets=50, n_bytes=2550, priority=655  35,d1,dst=01:80:c2:00:00:0e,d1_type=0x80cc actions=CONTROLLER:65535  cookie=0x0, duration=23.864s, table=0, n_packets=2, n_bytes=102, priority=0 actions=CONTROLLER:65535</pre>	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s3 --protocols=OpenFlow13 OFPST_FLOW reply (OF1.3) (xid=0x2):  cookie=0x0, duration=54.665s, table=0, n_packets=121, n_bytes=6171, priority=65  35,d1,dst=01:80:c2:00:00:0e,d1_type=0x80cc actions=CONTROLLER:65535  cookie=0x0, duration=33.903s, table=0, n_packets=2, n_bytes=84, priority=1,arp,  in_port=1,d1,dst=10:00:00:00:00:02,d1_dst=10:00:00:00:00:04 actions=output:2  cookie=0x0, duration=28.796s, table=0, n_packets=1, n_bytes=42, priority=1,arp,  in_port=2,d1,dst=10:00:00:00:00:04,d1_dst=10:00:00:00:00:03 actions=output:1  cookie=0x0, duration=31.894s, table=0, n_packets=1, n_bytes=80, priority=1,icmp,  in_port=1,d1,dst=10:00:00:00:00:03,d1_dst=10:00:00:00:00:04 actions=output:2  cookie=0x0, duration=54.671s, table=0, n_packets=20, n_bytes=1857, priority=0 actions=CONTROLLER:65535</pre>

- (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	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s1 --protocols=OpenFlow13 OFPST_FLOW reply (OF1.3) (xid=0x2):  cookie=0x0, duration=9.951s, table=0, n_packets=24, n_bytes=1224, priority=6553  5,d1,dst=01:80:c2:00:00:0e,d1_type=0x80cc actions=CONTROLLER:65535  cookie=0x0, duration=9.954s, table=0, n_packets=5, n_bytes=616, priority=0 actions=CONTROLLER:65535</pre>	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s1 --protocols=OpenFlow13 OFPST_FLOW reply (OF1.3) (xid=0x2):  cookie=0x0, duration=18.021s, table=0, n_packets=182, n_bytes=15402, priority=  65535,d1,dst=01:80:c2:00:00:0e,d1_type=0x80cc actions=CONTROLLER:65535  cookie=0x0, duration=37.032s, table=0, n_packets=2, n_bytes=84, priority=1,arp,  in_port=2,d1,dst=10:00:00:00:00:02,d1_dst=10:00:00:00:00:04 actions=output:3  cookie=0x0, duration=31.778s, table=0, n_packets=1, n_bytes=42, priority=1,arp,  in_port=3,d1,dst=10:00:00:00:00:04,d1_dst=10:00:00:00:00:02 actions=output:2  cookie=0x0, duration=37.025s, table=0, n_packets=1456639, n_bytes=6448290534,  priority=1,arp,in_port=2,d1,dst=10:00:00:00:00:04,d1_dst=10:00:00:00:00:02,tp_ds  8080 actions=output:2  cookie=0x0, duration=138.024s, table=0, n_packets=20, n_bytes=2090, priority=0  actions=CONTROLLER:65535</pre>
S3	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s3 --protocols=OpenFlow13 OFPST_FLOW reply (OF1.3) (xid=0x2):  cookie=0x0, duration=40.017s, table=0, n_packets=86, n_bytes=4386, priority=655  35,d1,dst=01:80:c2:00:00:0e,d1_type=0x80cc actions=CONTROLLER:65535  cookie=0x0, duration=40.022s, table=0, n_packets=9, n_bytes=1143, priority=0 actions=CONTROLLER:65535</pre>	<p>TCP packet will be sent back to h4 from h2, thus s3's flow table is modified</p> <pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s3 --protocols=OpenFlow13 OFPST_FLOW reply (OF1.3) (xid=0x2):  cookie=0x0, duration=172.188s, table=0, n_packets=375, n_bytes=19125, priority=  65535,d1,dst=01:80:c2:00:00:0e,d1_type=0x80cc actions=CONTROLLER:65535  cookie=0x0, duration=71.189s, table=0, n_packets=1131494, n_bytes=7467808, pri  ority=1,tcp,in_port=3,d1,dst=10:00:00:00:00:02,d1_dst=10:00:00:00:00:04,tp_dst=5  4780 actions=output:2  cookie=0x0, duration=172.193s, table=0, n_packets=18, n_bytes=2015, priority=0  actions=CONTROLLER:65535</pre>
	Generates TCP traffic	Receives TCP traffic
Mininet or hosts	<pre>root@qzy-VirtualBox:/media/sf_Data_Center_and_Cloud_Computing/lab3# 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 53194 connected with 10.0.0.2 port 8080 [ ID] Interval Transfer Bandwidth [ 23] 0.0-10.0 sec 64.4 CBytes 55.3 Gbits/sec root@qzy-VirtualBox:/media/sf_Data_Center_and_Cloud_Computing/lab3#</pre>	<pre>root@qzy-VirtualBox:/media/sf_Data_Center_and_Cloud_Computing/lab3# 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 53194 [ ID] Interval Transfer Bandwidth [ 24] 0.0-10.0 sec 64.4 CBytes 55.3 Gbits/sec</pre>

- (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	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s1 --protocols=OpenFlow13 OpenFlow reply (OF1.3) (xid=0x2):  cookie=0x0, duration=138.021s, table=0, n_packets=302, n_bytes=15402, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x8b8c actions=CONTROLLER:65535  cookie=0x0, duration=37.032s, table=0, n_packets=2, n_bytes=84, priority=1,arp,in_port=2,dl_src=10:00:00:00:00:02,dl_dst=10:00:00:00:00:04 actions=output:3  cookie=0x0, duration=31.778s, table=0, n_packets=1, n_bytes=42, priority=1,arp,in_port=3,dl_src=10:00:00:00:00:04,dl_dst=10:00:00:00:00:02 actions=output:2  cookie=0x0, duration=37.028s, table=0, n_packets=135639, n_bytes=6448290334, priority=1,tcp,in_port=3,dl_src=10:00:00:00:00:04,dl_dst=10:00:00:00:00:02,tp_dst=8080 actions=output:2  cookie=0x0, duration=138.024s, table=0, n_packets=20, n_bytes=2090, priority=0 actions=CONTROLLER:65535</pre>	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s1 --protocols=OpenFlow13 OpenFlow reply (OF1.3) (xid=0x2):  cookie=0x0, duration=352.077s, table=0, n_packets=770, n_bytes=39270, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x8b8c actions=CONTROLLER:65535  cookie=0x0, duration=251.084s, table=0, n_packets=4, n_bytes=168, priority=1,arp,in_port=2,dl_src=10:00:00:00:00:02,dl_dst=10:00:00:00:00:04 actions=output:3  cookie=0x0, duration=245.830s, table=0, n_packets=3, n_bytes=126, priority=1,arp,in_port=3,dl_src=10:00:00:00:00:04,dl_dst=10:00:00:00:00:02 actions=output:2  cookie=0x0, duration=251.077s, table=0, n_packets=145639, n_bytes=6448290334, priority=1,tcp,in_port=3,dl_src=10:00:00:00:00:04,dl_dst=10:00:00:00:00:02,tp_dst=8080 actions=output:2  cookie=0x0, duration=40.153s, table=0, n_packets=903, n_bytes=1365336, priority=1,udp,in_port=2,dl_src=10:00:00:00:00:02,dl_dst=10:00:00:00:00:04 actions=output:3  cookie=0x0, duration=352.076s, table=0, n_packets=26, n_bytes=4218, priority=0 actions=CONTROLLER:65535</pre>
S3	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s3 --protocols=OpenFlow13 OpenFlow reply (OF1.3) (xid=0x2):  cookie=0x0, duration=172.188s, table=0, n_packets=375, n_bytes=19125, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x8b8c actions=CONTROLLER:65535  cookie=0x0, duration=71.189s, table=0, n_packets=1131494, n_bytes=74678780, priority=1,tcp,in_port=3,dl_src=10:00:00:00:00:02,dl_dst=10:00:00:00:00:04,tp_dst=54790 actions=output:2  cookie=0x0, duration=172.193s, table=0, n_packets=18, n_bytes=2015, priority=0 actions=CONTROLLER:65535</pre>	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s3 --protocols=OpenFlow13 OpenFlow reply (OF1.3) (xid=0x2):  cookie=0x0, duration=405.049s, table=0, n_packets=884, n_bytes=45084, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x8b8c actions=CONTROLLER:65535  cookie=0x0, duration=304.050s, table=0, n_packets=1131494, n_bytes=74678780, priority=1,tcp,in_port=3,dl_src=10:00:00:00:00:02,dl_dst=10:00:00:00:00:04,tp_dst=54790 actions=output:2  cookie=0x0, duration=405.054s, table=0, n_packets=23, n_bytes=2631, priority=0 actions=CONTROLLER:65535</pre>
Mininet or hosts	Generates UDP traffic iperf in receiver will send upd packet back to the sender to ack, but h4 is not allowed to send udp packet. <pre>root@qzy-VirtualBox:/media/sf_Data_Center_and_Cloud_Computing/lab3# iperf -c 10.0.0.4 -p 8080 -u Client connecting to 10.0.0.4, UDP port 8080 Sending 1470 byte datagrams UDP buffer size: 208 KByte (default)  [ 23] local 10.0.0.2 port 34622 connected with 10.0.0.4 port 8080 [ 10] Interval Transfer Bandwidth [ 23] 0.0-10.0 sec 1.25 MBytes 1.05 Mbits/sec [ 23] Sent 993 datagrams [ 23] WARNING: did not receive ack of last datagram after 10 tries.</pre>	Receives UDP traffic <pre>root@qzy-VirtualBox:/media/sf_Data_Center_and_Cloud_Computing/lab3# iperf -s -p 8080 -u Server listening on UDP port 8080 Receiving 1470 byte datagrams UDP buffer size: 208 KByte (default)  [ 23] local 10.0.0.4 port 8080 connected with 10.0.0.2 port 34622 [ 10] Interval Transfer Bandwidth Jitter Lost/Total Datagrams [ 23] 0.0-10.0 sec 1.25 MBytes 1.05 Mbits/sec 0.036 ms 0/ 935 (0%)</pre>

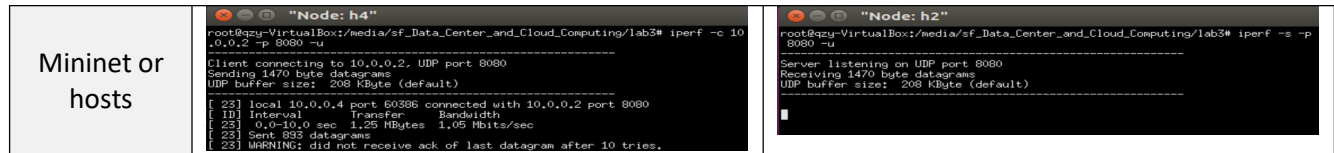
- (f) Generate HTTP traffic from **H2 to H1**, and take **screenshots** of the flow table on S2 before and after the flow is generated. (ovs-ofctl dump-flows) Also, the screenshot of your Mininet or host that generates/receives the HTTP traffic.

	Before HTTP flow is generated	After HTTP flow is generated
S2	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s2 --protocols=OpenFlow13 OpenFlow reply (OF1.3) (xid=0x2):  cookie=0x0, duration=3.243s, table=0, n_packets=0, n_bytes=306, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x8b8c actions=CONTROLLER:65535  cookie=0x0, duration=3.250s, table=0, n_packets=2, n_bytes=102, priority=0 actions=CONTROLLER:65535</pre>	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s2 --protocols=OpenFlow13 OpenFlow reply (OF1.3) (xid=0x2):  cookie=0x0, duration=74.038s, table=0, n_packets=161, n_bytes=8211, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x8b8c actions=CONTROLLER:65535  cookie=0x0, duration=12.931s, table=0, n_packets=1, n_bytes=42, priority=5,tcp,nw_src=10.0.0.2,tp_dst=80 actions=CONTROLLER:65509  cookie=0x0, duration=12.931s, table=0, n_packets=1, n_bytes=42, priority=1,arp,in_port=3,dl_src=10:00:00:00:00:01,dl_dst=10:00:00:00:00:02 actions=output:1  cookie=0x0, duration=74.045s, table=0, n_packets=20, n_bytes=2108, priority=0 actions=CONTROLLER:65535</pre>
Mininet or hosts	Generates HTTP traffic <pre>root@qzy-VirtualBox:/media/sf_Data_Center_and_Cloud_Computing/lab3# iperf -c 10.0.0.1 -p 80 connect Failed: Connection refused</pre>	Receives HTTP traffic <pre>root@qzy-VirtualBox:/media/sf_Data_Center_and_Cloud_Computing/lab3# iperf -s -p 80 Server listening on TCP port 80 TCP window size: 65.5 KByte (default)  []</pre>

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	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s4 --protocols=OpenFlow13 OpenFlow reply (OF1.3) (xid=0x2):  cookie=0x0, duration=140.906s, table=0, n_packets=310, n_bytes=15810, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x8b8c actions=CONTROLLER:65535  cookie=0x0, duration=140.908s, table=0, n_packets=22, n_bytes=2548, priority=0 actions=CONTROLLER:65535</pre>	<pre>root@qzy-VirtualBox:/media# ovs-ofctl dump-flows s4 --protocols=OpenFlow13 OpenFlow reply (OF1.3) (xid=0x2):  cookie=0x0, duration=253.352s, table=0, n_packets=556, n_bytes=28350, priority=65535,dl_dst=01:80:c2:00:00:0e,dl_type=0x8b8c actions=CONTROLLER:65535  cookie=0x0, duration=34.015s, table=0, n_packets=903, n_bytes=1365336, priority=5,udp,nw_src=10.0.0.4 actions=clear_actions  cookie=0x0, duration=253.354s, table=0, n_packets=28, n_bytes=4648, priority=0 actions=CONTROLLER:65535</pre>
	Generates UDP traffic	Receives UDP traffic



(h) Please find what is “Spanning Tree” and “Spanning Tree Protocol”? What’s the purpose of the protocol?

Spanning tree is a connected acyclic undirected graph, and is the sub-graph of the the original graph that contains all the vertex.

Spanning tree protocol helps to find the spanning tree of a ethernet network and turn down all other links not within the spanning tree, to avoid broadcast storm and provide failure recovery.

The purpose of this protocol is to create a loop free ethernet network, so as to avoid broadcast storm triggered by MAC address look up.

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

It is unnecessary to implement STP in SDN, because we can have control plane to instruct the switch how to forward the packets manually, but STP is still useful in SDN.

(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?

The STP in SDN network will be implemented by the controller. The controller can first probe the layout of all the switches in the network, and then it can determine the spanning tree and instruct the switches to forward the broadcast packets only within this spanning tree. However, links not within the spanning tree can still be used to transmit packets that are not broadcast, unlike the traditional STP, where those links are invalidated.

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

1. SDN allows fine-grain central control over the network functionality.
2. SDN allows network setting automation.
3. SDN’s central controlling makes the network more efficient and guarantees content delivery.

(l) Include the controller’s code.

(Upload with your report or attach a sharable link)

Upload together.

(m) Include the topology file

(Upload with your report or attach a sharable link)

Upload together.

(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. ARP proxy and avoid broadcast storm
2. Find shortest path using networkx, with --observe-links, and when there are multiple paths, choose the one that meet the need.
3. Set up the flow table to drop specific UDP packet simply and send specific TCP packet to controller to

generate a TCP\_RST.

4. Match packets with ARP, ICMP, UDP, TCP and HTTP specifically.

5. Much higher work load than previous lab.

6. Caught a cold during these days.

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