

Lab 2 Report: SDN Simulation

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

Name: Shengwei Huang ID: Sh6203 Date: 10/16/2020

1. Objectives

- Fully understand the operation of Openflow and observe the operations
- Master the simulation tool mininet

2. Equipment Needs

- Computers
- Internet

3. Experiments

3.1 Basics

1. Install Ubuntu on your computer, you can install it on a VM using either VMWare player or VirtualBox:
VMWare:

https://my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/6_0

VirtualBox:

<https://www.virtualbox.org/wiki/Downloads>

Ubuntu 16 or 18 is recommended! Ubuntu 20 might have some issues with python 2 and mininet!!

2. Follow the instructions to set up Mininet on your Ubuntu VM:

<http://mininet.org/download/>

3. Perform basic simulations following these steps:

<http://mininet.org/walkthrough>

3.2 Openflow

Part 1. Observe Openflow control messages:

1. Launch Mininet with default controller (**No “--controller remote” !**):

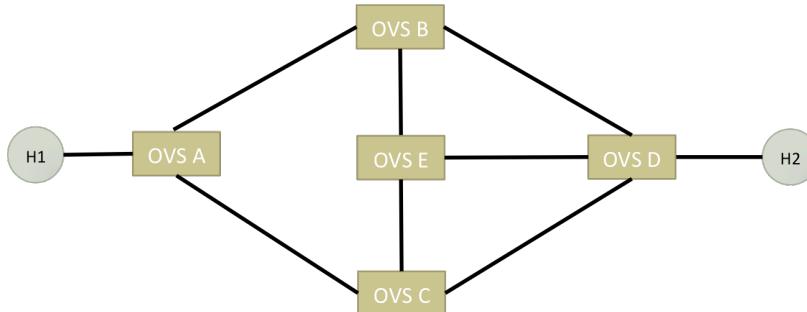
```
$ sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo
```

2. Install and open Wireshark in **Ubuntu**. Start listening to all the interfaces.

3. In Mininet:

```
mininet> pingall
```

You will see successful connections. Stop the Wireshark and find Openflow control flows. Fill in the table 4(a)



Part 2. Manually install flow entries in the OVS's

1. Use Mininet to create the topology above, where A, B, C, D, and E are all Openflow switches.
(Remember to add “--controller remote” to disable default controller)

2. Enforce the following policies so that,

- a. Traffic from H1 → H2
 - i. HTTP traffic with d_port=80 follows path: A-B-D
 - ii. other traffic follows path: A-C-E-D
- b. Traffic from H2 → H1
 - i. HTTP traffic with s_port=80, follow path: D-C-A
 - ii. other traffic, follow path: D-B-E-C-A

(i.e., you need to define a good match-action for the switches to achieve these.)

- c. verify your policies by,
 - i. generating corresponding traffic (**using network debugging tools**)
 - ii. capturing packets with Wireshark (listen to some switches' interfaces to see if the packets pass by)

You can use OVS-OFCTL to manually install rules on switches (preferred method), or you can install a simple controller using RYU/POX/NOX/Beacon (Not recommended for lab 2 you'll do it in lab 3).

Part 3. Write a program that automatically creates a fat-tree with a given size N

1. Using mininet, create a 2-stage Fat Tree network using N-port switches, where N is a variable that should be an even number. (You don't need to check the connectivity, just create the topology.)

(Set N as a global variable in the topology (python) file)

4. Reports

(a) A screenshot of OpenFlow control messages you captured with Wireshark.

The Wireshark interface displays two windows of captured network traffic:

- Top Window:** Shows a list of 79 captured packets. The columns include No., Time, Source, Destination, Protocol, Length, and Info. Most packets are of type OFPT_PACKET_IN or OFPT_PACKET_OUT between localhost and localhost. One packet is an OFPT_FLOW_MOD message.
- Bottom Window:** Provides a detailed view of a selected packet (Frame 4). It shows the raw hex dump (0000 to 00b0), the corresponding ASCII representation, and the F(W-) frame boundary indicator.
- Protocol View:** Below the main windows, the "OpenFlow 1.3: Protocol" tab is selected, showing a list of 131 packets. This tab lists the same information as the main list but includes a "Packets: 131 · Displayed: 40 (30.5%)" status bar.

Protocol Details:

- OFPT_PACKET_IN:** 180 bytes on wire (1440 bits), 180 bytes captured (1440 bits) on interface any, id 0
- OFPT_PACKET_OUT:** 108 bytes on wire (864 bits), 108 bytes captured (864 bits) on interface any, id 0
- OFPT_FLOW_MOD:** 212 bytes on wire (1696 bits), 212 bytes captured (1696 bits) on interface any, id 0

Selected Packet Details:

```

Frame 4: 180 bytes on wire (1440 bits), 180 bytes captured (1440 bits) on interface any, id 0
  ▶ Linux cooked capture
  ▶ Internet Protocol Version 4, Src: localhost (127.0.0.1), Dst: localhost (127.0.0.1)
  ▶ Transmission Control Protocol, Src Port: 52338, Dst Port: 6633, Seq: 1, Ack: 1, Len: 112
  ▶ OpenFlow 1.3

0000  00 00 03 04 00 06 00 00 00 00 00 00 00 00 00 00
0010  45 c0 00 a4 2d 19 40 00 40 06 79 7f 00 00 01 E...-@ @-y...
0020  7f 00 00 01 cc 72 19 e9 16 80 7b 15 56 37 cd 83 ..r...-{V7...
0030  80 18 00 56 fe 98 00 00 01 01 08 0a e7 c7 ce cf ..V...&...
0040  e7 c7 ca c9 04 00 00 70 00 00 00 00 00 01 00 0c .F...p.....&
0050  00 46 00 00 00 00 00 00 00 00 00 00 00 00 01 00 0c .F.....33
0060  80 00 00 04 00 00 00 01 00 00 00 00 00 00 00 00 33 33
0070  00 00 00 02 22 5f 46 28 57 c4 86 dd 60 00 00 00 ..._F( W...
0080  00 10 3a ff fe 80 00 00 00 00 00 00 20 5f 46 ff .:....F...
0090  fe 28 57 c4 ff 02 00 00 00 00 00 00 00 00 01 01 22 5f .(W...._
00a0  00 00 00 02 85 00 fe 96 00 00 00 00 01 01 22 5f ....."-_
00b0  46 28 57 c4 F(W-

```

(b)

Commands	Meaning of the commands
ovs-ofctl show s1	Show the current state of the openflow switch s1. Including the information of the flow table and ports.
ovs-ofctl dump-flows s1	Print all entries of flow which is in the table of switch that match flows.
ovs-ofctl add-flow s1 in_port=1,actions=output:2	Add flow entry to the table of switch. And also match the OpenFlow port is 1. Define the output as an action and the port number is 2.
ovs-ofctl add-flow s1 priority=500,in_port=1,dl_type=0x0800,nw_proto=6, actions=output:2	(what are the matches and what is the action?) Add flow entry to the table of switch. And also set the priority is 500 which means set the order of running flow order. Also match the OpenFlow port is 1. Also dl_type=0x0800 means to match all ethernet type 0x0800 which is IPv4 packets. And nw_proto=6 means to match all TCP packets. dl_type=0x0800,nw_proto=6 are the same as TCP. Also Define the output as an action and the port number is 2.
Question: What is “priority”? Why do we need “Priority”? What is the default priority if the priority is not given?	
1. When traffic goes through from one host to another one. A wildcarded entry will match the priority to compare one to another.	
2. Because there should be more than one flows to go through from one host to another. And then we need priority to set the highest priority. And the traffic will always go through the highest priority first.	
3. Default priority: 32768	
Question: What is the default priority if the priority is not given?	
32768	

(c) Output of mininet “net” command for both topologies (part 2 & part 3). (you can use any N for Fat-tree, ex: 4, 6, 8)

```
Part2:
[mininet> net
h1 h1-eth1:s1-eth1
h2 h2-eth1:s5-eth4
s1 lo:  s1-eth1:h1-eth1 s1-eth2:s2-eth1 s1-eth3:s3-eth1
s2 lo:  s2-eth1:s1-eth2 s2-eth2:s4-eth1 s2-eth3:s5-eth1
s3 lo:  s3-eth1:s1-eth3 s3-eth2:s4-eth2 s3-eth3:s5-eth3
s4 lo:  s4-eth1:s2-eth2 s4-eth2:s3-eth2 s4-eth3:s5-eth2
s5 lo:  s5-eth1:s2-eth3 s5-eth2:s4-eth3 s5-eth3:s3-eth3 s5-eth4:h2-eth
c0  -
```

Part3:

```

mininet> net
host1 host1-eth0:switch1-eth1 host1-eth1:switch2-eth1 host1-eth2:switch3-eth1 host1-eth3:switch4-eth1 host1-eth4:switch5-eth1 host1-eth5:switch6-eth1
host2 host2-eth0:switch1-eth2 host2-eth1:switch2-eth2 host2-eth2:switch3-eth2 host2-eth3:switch4-eth2 host2-eth4:switch5-eth2 host2-eth5:switch6-eth2
host3 host3-eth0:switch1-eth3 host3-eth1:switch2-eth3 host3-eth2:switch3-eth3 host3-eth3:switch4-eth3 host3-eth4:switch5-eth3 host3-eth5:switch6-eth3
host4 host4-eth0:switch1-eth4 host4-eth1:switch2-eth4 host4-eth2:switch3-eth4 host4-eth3:switch4-eth4 host4-eth4:switch5-eth4 host4-eth5:switch6-eth4
host5 host5-eth0:switch1-eth5 host5-eth1:switch2-eth5 host5-eth2:switch3-eth5 host5-eth3:switch4-eth5 host5-eth4:switch5-eth5 host5-eth5:switch6-eth5
host6 host6-eth0:switch1-eth6 host6-eth1:switch2-eth6 host6-eth2:switch3-eth6 host6-eth3:switch4-eth6 host6-eth4:switch5-eth6 host6-eth5:switch6-eth6
host7 host7-eth0:switch1-eth7 host7-eth1:switch2-eth7 host7-eth2:switch3-eth7 host7-eth3:switch4-eth7 host7-eth4:switch5-eth7 host7-eth5:switch6-eth7
host8 host8-eth0:switch1-eth8 host8-eth1:switch2-eth8 host8-eth2:switch3-eth8 host8-eth3:switch4-eth8 host8-eth4:switch5-eth8 host8-eth5:switch6-eth8
host9 host9-eth0:switch1-eth9 host9-eth1:switch2-eth9 host9-eth2:switch3-eth9 host9-eth3:switch4-eth9 host9-eth4:switch5-eth9 host9-eth5:switch6-eth9
host10 host10-eth0:switch1-eth10 host10-eth1:switch2-eth10 host10-eth2:switch3-eth10 host10-eth3:switch4-eth10 host10-eth4:switch5-eth10 host10-eth5:switch6-eth10
host11 host11-eth0:switch1-eth11 host11-eth1:switch2-eth11 host11-eth2:switch3-eth11 host11-eth3:switch4-eth11 host11-eth4:switch5-eth11 host11-eth5:switch6-eth11
host12 host12-eth0:switch1-eth12 host12-eth1:switch2-eth12 host12-eth2:switch3-eth12 host12-eth3:switch4-eth12 host12-eth4:switch5-eth12 host12-eth5:switch6-eth12
host13 host13-eth0:switch1-eth13 host13-eth1:switch2-eth13 host13-eth2:switch3-eth13 host13-eth3:switch4-eth13 host13-eth4:switch5-eth13 host13-eth5:switch6-eth13
host14 host14-eth0:switch1-eth14 host14-eth1:switch2-eth14 host14-eth2:switch3-eth14 host14-eth3:switch4-eth14 host14-eth4:switch5-eth14 host14-eth5:switch6-eth14
host15 host15-eth0:switch1-eth15 host15-eth1:switch2-eth15 host15-eth2:switch3-eth15 host15-eth3:switch4-eth15 host15-eth4:switch5-eth15 host15-eth5:switch6-eth15
host16 host16-eth0:switch1-eth16 host16-eth1:switch2-eth16 host16-eth2:switch3-eth16 host16-eth3:switch4-eth16 host16-eth4:switch5-eth16 host16-eth5:switch6-eth16
host17 host17-eth0:switch1-eth17 host17-eth1:switch2-eth17 host17-eth2:switch3-eth17 host17-eth3:switch4-eth17 host17-eth4:switch5-eth17 host17-eth5:switch6-eth17
host18 host18-eth0:switch1-eth18 host18-eth1:switch2-eth18 host18-eth2:switch3-eth18 host18-eth3:switch4-eth18 host18-eth4:switch5-eth18 host18-eth5:switch6-eth18
core1 lo: core1-eth1:switch1-eth19 core1-eth2:switch2-eth19 core1-eth3:switch3-eth19 core1-eth4:switch4-eth19 core1-eth5:switch5-eth19 core1-eth6:switch6-eth19
core2 lo: core2-eth1:switch1-eth20 core2-eth2:switch2-eth20 core2-eth3:switch3-eth20 core2-eth4:switch4-eth20 core2-eth5:switch5-eth20 core2-eth6:switch6-eth20
core3 lo: core3-eth1:switch1-eth21 core3-eth2:switch2-eth21 core3-eth3:switch3-eth21 core3-eth4:switch4-eth21 core3-eth5:switch5-eth21 core3-eth6:switch6-eth21
switch1 lo: switch1-eth1:host1-eth0 switch1-eth2:host0-eth0 switch1-eth3:host3-eth0 switch1-eth4:host4-eth0 switch1-eth5:host5-eth0 switch1-eth6:host6-eth0 switch1-eth7:host7-eth0 switch1-eth8:host8-eth0 switch1-eth9:host9-eth0 switch1-eth10:eth0 switch1-eth11:host11-eth0 switch1-eth12:host12-eth0 switch1-eth13:host13-eth0 switch1-eth14:host14-eth0 switch1-eth15:host15-eth0 switch1-eth16:host16-eth0
switch2 lo: switch2-eth1:host1-eth1 switch2-eth2:host1-eth1 switch2-eth3:host3-eth1 switch2-eth4:host4-eth1 switch2-eth5:host5-eth1 switch2-eth6:host6-eth1 switch2-eth7:host7-eth1 switch2-eth8:host8-eth1 switch2-eth9:host9-eth1 switch2-eth10:eth1 switch2-eth11:host11-eth1 switch2-eth12:host12-eth1 switch2-eth13:host13-eth1 switch2-eth14:host14-eth1 switch2-eth15:host15-eth1 switch2-eth16:host16-eth1
h1 switch2-eth17:host17-eth1 switch2-eth18:host18-eth1 switch2-eth19:host19-eth1 switch2-eth20:core2-eth2 switch2-eth21:core3-eth2
switch3 lo: switch3-eth1:host1-eth2 switch3-eth2:host2-eth2 switch3-eth3:host3-eth2 switch3-eth4:host4-eth2 switch3-eth5:host5-eth2 switch3-eth6:host6-eth2 switch3-eth7:host7-eth2 switch3-eth8:host8-eth2 switch3-eth9:host9-eth2 switch3-eth10:eth2 switch3-eth11:host11-eth2 switch3-eth12:host12-eth2 switch3-eth13:host13-eth2 switch3-eth14:host14-eth2 switch3-eth15:host15-eth2 switch3-eth16:host16-eth2
h2 switch3-eth17:host17-eth2 switch3-eth18:host18-eth2 switch3-eth19:host19-eth2 switch3-eth20:core2-eth3 switch3-eth21:core3-eth3
switch4 lo: switch4-eth1:host1-eth3 switch4-eth2:host2-eth3 switch4-eth3:host3-eth3 switch4-eth4:host4-eth3 switch4-eth5:host5-eth3 switch4-eth6:host6-eth3 switch4-eth7:host7-eth3 switch4-eth8:host8-eth3 switch4-eth9:host9-eth3 switch4-eth10:eth3 switch4-eth11:host11-eth3 switch4-eth12:host12-eth3 switch4-eth13:host13-eth3 switch4-eth14:host14-eth3 switch4-eth15:host15-eth3 switch4-eth16:host16-eth3
h3 switch4-eth17:host17-eth3 switch4-eth18:host18-eth3 switch4-eth19:host19-eth3 switch4-eth20:core2-eth4 switch4-eth21:core3-eth4
switch5 lo: switch5-eth1:host1-eth4 switch5-eth2:host2-eth4 switch5-eth3:host3-eth4 switch5-eth4:host4-eth4 switch5-eth5:host5-eth4 switch5-eth6:host6-eth4 switch5-eth7:host7-eth4 switch5-eth8:host8-eth4 switch5-eth9:host9-eth4 switch5-eth10:eth4 switch5-eth11:host11-eth4 switch5-eth12:host12-eth4 switch5-eth13:host13-eth4 switch5-eth14:host14-eth4 switch5-eth15:host15-eth4 switch5-eth16:host16-eth4
h4 switch5-eth17:host17-eth4 switch5-eth18:host18-eth4 switch5-eth19:core1-eth5 switch5-eth20:core2-eth5 switch5-eth21:core3-eth5
switch6 lo: switch6-eth1:host1-eth5 switch6-eth2:host2-eth5 switch6-eth3:host3-eth5 switch6-eth4:host4-eth5 switch6-eth5:host5-eth5 switch6-eth6:host6-eth5 switch6-eth7:host7-eth5 switch6-eth8:host8-eth5 switch6-eth9:host9-eth5 switch6-eth10:eth5 switch6-eth11:host11-eth5 switch6-eth12:host12-eth5 switch6-eth13:host13-eth5 switch6-eth14:host14-eth5 switch6-eth15:host15-eth5 switch6-eth16:host16-eth5
h5 switch6-eth17:host17-eth5 switch6-eth18:host18-eth5 switch6-eth19:core1-eth6 switch6-eth20:core2-eth6 switch6-eth21:core3-eth6
c0
mininet>

```

- (d) The command you use to start your Mininet topology and the screen shot of the Mininet output while creating the networks.

Command you use to start your Mininet topology:

(Remember to add “**--controller remote**” to disable default controller; otherwise, you’ll get zero for the following tasks.)

Part2:

```
sudo mn --custom topo-5sw-2host.py --topo mytopo --controller remote
```

```
[sh6203@sh6203-VirtualBox:~$ sudo mn --custom topo-5sw-2host.py --topo mytopo --controller remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (s1, s2) (s1, s3) (s2, s4) (s2, s5) (s3, s4) (s4, s5) (s5, h2) (s5, s3)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
mininet>
```

Part3:

```
sh6203@sh6203-VirtualBox:~$ sudo mn --custom fatTree.py --topo mytopo --controller remote
[sudo] password for sh6203:
port number: 6633
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
host1 host2 host3 host4 host5 host6 host7 host8 host9 host10 host11 host12 host13 host14 host15 host16 host17 host18
*** Adding switches:
core1 core2 core3 switch1 switch2 switch3 switch4 switch5 switch6
*** Adding links:
(core1, switch1) (core1, switch2) (core1, switch3) (core1, switch4) (core1, switch5) (core2, switch6) (core2, switch1) (core2, switch2) (core2, switch3) (core2, switch4) (core2, switch5) (core2, switch6) (core3, switch1) (core3, switch2) (core3, switch3) (core3, switch4) (core3, switch5) (core3, switch6) (host1, switch1) (host1, switch2) (host1, switch3) (host1, switch4) (host1, switch5) (host1, switch6) (host2, switch1) (host2, switch2) (host2, switch3) (host2, switch4) (host2, switch5) (host2, switch6) (host3, switch1) (host3, switch2) (host3, switch3) (host3, switch4) (host3, switch5) (host3, switch6) (host4, switch1) (host4, switch2) (host4, switch3) (host4, switch4) (host4, switch5) (host4, switch6) (host5, switch1) (host5, switch2) (host5, switch3) (host5, switch4) (host5, switch5) (host5, switch6) (host6, switch1) (host6, switch2) (host6, switch3) (host6, switch4) (host6, switch5) (host6, switch6) (host7, switch1) (host7, switch2) (host7, switch3) (host7, switch4) (host7, switch5) (host7, switch6) (host8, switch1) (host8, switch2) (host8, switch3) (host8, switch4) (host8, switch5) (host8, switch6) (host9, switch1) (host9, switch2) (host9, switch3) (host9, switch4) (host9, switch5) (host9, switch6) (host10, switch1) (host10, switch2) (host10, switch3) (host10, switch4) (host10, switch5) (host10, switch6) (host11, switch1) (host11, switch2) (host11, switch3) (host11, switch4) (host11, switch5) (host11, switch6) (host12, switch1) (host12, switch2) (host12, switch3) (host12, switch4) (host12, switch5) (host12, switch6) (host13, switch1) (host13, switch2) (host13, switch3) (host13, switch4) (host13, switch5) (host13, switch6) (host14, switch1) (host14, switch2) (host14, switch3) (host14, switch4) (host14, switch5) (host14, switch6) (host15, switch1) (host15, switch2) (host15, switch3) (host15, switch4) (host15, switch5) (host15, switch6) (host16, switch1) (host16, switch2) (host16, switch3) (host16, switch4) (host16, switch5) (host16, switch6) (host17, switch1) (host17, switch2) (host17, switch3) (host17, switch4) (host17, switch5) (host17, switch6) (host18, switch1) (host18, switch2) (host18, switch3) (host18, switch4) (host18, switch5) (host18, switch6)
*** Configuring hosts
host1 host2 host3 host4 host5 host6 host7 host8 host9 host10 host11 host12 host13 host14 host15 host16 host17 host18
*** Starting controller
c0
*** Starting 9 switches
core1 core2 core3 switch1 switch2 switch3 switch4 switch5 switch6 ...
*** Starting CLI:
```

- (e) Briefly explain how you produce different traffic to verify whether the rules installed function correctly.

(Hint: Use the network debugging tool in lab 1)

After I finish different traffic, I use pingall to test whether I have finished correctly. And it shows

```
[mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
```

It can be from host1 to host2 and from host2 to host1, which means that my rules can be used to create flows. Also I use xterm h1 in mininet to open node h1. And use hping3 -p 80 -c 1 10.0.0.2 to hping host2. To check whether they can send and receive packets successfully.

```
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (s1, s2) (s1, s3) (s1, s4) (s1, s5)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
mininet> xterm h1
mininet> 
```

Here it shows 1 packets transmitted and 1 packets received. Also there are 0% packet loss. So the packet can be transferred from the flow.

And also I use sudo ovs-ofctl dump-flows s1, sudo ovs-ofctl dump-flows s2, sudo ovs-ofctl dump-flows s3, sudo ovs-ofctl dump-flows s4, sudo ovs-ofctl dump-flows s5, to show all the process information of my 5 switches, to see whether they are transferring the packets correctly.

```
[sh6203@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=250.745s, table=0, n_packets=1, n_bytes=54, idle_age=180, priority=100
,tcp,in_port=1,tp_dst=80 actions=output:2
cookie=0x0, duration=250.678s, table=0, n_packets=1, n_bytes=54, idle_age=180, priority=100
,tcp,in_port=3,tp_src=80 actions=output:1
cookie=0x0, duration=250.721s, table=0, n_packets=7, n_bytes=434, idle_age=5, priority=0,in
_port=1 actions=output:3
cookie=0x0, duration=250.646s, table=0, n_packets=45, n_bytes=5754, idle_age=5, priority=0,
in_port=3 actions=output:1
[sh6203@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s2
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=276.805s, table=0, n_packets=1, n_bytes=54, idle_age=206, priority=100
,tcp,in_port=1,tp_dst=80 actions=output:3
cookie=0x0, duration=276.732s, table=0, n_packets=17, n_bytes=1799, idle_age=31, priority=0
,in_port=3 actions=output:2
[sh6203@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=279.136s, table=0, n_packets=1, n_bytes=54, idle_age=208, priority=100
,tcp,in_port=3,tp_src=80 actions=output:1
cookie=0x0, duration=279.165s, table=0, n_packets=17, n_bytes=1799, idle_age=33, priority=0
,in_port=1 actions=output:2
cookie=0x0, duration=279.104s, table=0, n_packets=37, n_bytes=4529, idle_age=11, priority=0
,in_port=2 actions=output:1
[sh6203@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s4
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=280.676s, table=0, n_packets=27, n_bytes=3164, idle_age=35, priority=0
,in_port=2 actions=output:3
cookie=0x0, duration=280.627s, table=0, n_packets=27, n_bytes=3164, idle_age=35, priority=0
,in_port=1 actions=output:2
[sh6203@sh6203-VirtualBox:~$ sudo ovs-ofctl dump-flows s5
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=282.404s, table=0, n_packets=1, n_bytes=54, idle_age=212, priority=100
,tcp,in_port=1,tp_dst=80 actions=output:4
cookie=0x0, duration=282.366s, table=0, n_packets=1, n_bytes=54, idle_age=212, priority=100
,tcp,in_port=4,tp_src=80 actions=output:3
cookie=0x0, duration=282.373s, table=0, n_packets=37, n_bytes=4529, idle_age=14, priority=0
,in_port=2 actions=output:4
cookie=0x0, duration=282.346s, table=0, n_packets=7, n_bytes=434, idle_age=49, priority=0,i
n_port=4 actions=output:1
It also shows my input port and output port.
```

- (f) With the produced traffic, show the screenshots of Wireshark capture on different links (switch with interface) to verify the paths taken by different traffic are correct.

Wireshark Screenshot 1 (Top):

Frame 36: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface s4-eth2, id 6

Ethernet II, Src: de:b0:ec:be:df:8a (de:b0:ec:be:df:8a), Dst: e2:fc:fe:95:25:5d (e2:fc:fe:95:25:5d)

Internet Protocol Version 4, Src: 10.0.0.1 (10.0.0.1), Dst: 10.0.0.2 (10.0.0.2)

Internet Control Message Protocol

No.	Time	Source	Destination	Protocol	Length	Info	Interface
5	-0.000260773	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x5c65, seq=1/256, ttl=64 (reply in 6)	s1-eth1
11	-0.000154915	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x5c65, seq=1/256, ttl=64 (reply in 12)	s1-eth3
17	-0.000152631	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x5c65, seq=1/256, ttl=64 (reply in 18)	s3-eth1
3	-0.000100035	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x5c65, seq=1/256, ttl=64 (reply in 4)	s3-eth2
33	-0.000097588	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x5c65, seq=1/256, ttl=64 (reply in 34)	s4-eth2
25	-0.000068391	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x5c65, seq=1/256, ttl=64 (reply in 27)	s4-eth3
15	-0.000065616	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x5c65, seq=1/256, ttl=64 (no response found...)	s5-eth2
29	-0.000039043	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id=0x5c65, seq=1/256, ttl=64 (reply in 30)	s5-eth4
30	-0.000027069	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5c65, seq=1/256, ttl=64 (request in 29)	s5-eth4
1	0.000000000	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5c65, seq=1/256, ttl=64	s5-eth1
22	0.000001856	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5c65, seq=1/256, ttl=64	s2-eth3
27	0.000027465	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5c65, seq=1/256, ttl=64 (request in 25)	s2-eth2
21	0.000029247	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5c65, seq=1/256, ttl=64	s4-eth1
34	0.000052820	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5c65, seq=1/256, ttl=64 (request in 33)	s4-eth2
4	0.000053628	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5c65, seq=1/256, ttl=64 (request in 3)	s3-eth2
18	0.000076093	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5c65, seq=1/256, ttl=64 (request in 17)	s3-eth1
12	0.000076882	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5c65, seq=1/256, ttl=64 (request in 11)	s1-eth3
6	0.000099406	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5c65, seq=1/256, ttl=64 (request in 5)	s1-eth1
31	0.001754694	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 32)	s5-eth4
2	0.001761306	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (no response found...)	s5-eth1
23	0.001762937	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (no response found...)	s2-eth3
28	0.001765483	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (no response found...)	s2-eth2
24	0.001766799	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 26)	s4-eth1
35	0.001768889	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 36)	s4-eth2
9	0.001770417	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 10)	s3-eth2
19	0.001772664	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 20)	s3-eth1
13	0.001773764	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 14)	s1-eth3
7	0.001775766	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 8)	s1-eth1
8	0.001782622	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 7)	s1-eth1
14	0.001784077	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 13)	s1-eth3
20	0.001784534	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 19)	s3-eth1
10	0.001785632	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 9)	s3-eth2
36	0.001785965	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 35)	s4-eth2
26	0.001787226	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 24)	s4-eth3
16	0.001788600	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64	s5-eth2
32	0.001790128	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 31)	s5-eth4
37	2.110848737	fe80::f829:80ff:fe...	ff02::2	ICMPv6	70	Router Solicitation from fa:29:80:a2:c9:6c	s3-eth3
38	2.110857244	fe80::f829:80ff:fe...	ff02::2	ICMPv6	70	Router Solicitation from fa:29:80:a2:c9:6c	s5-eth3

Frame 31: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface s5-eth4, id 15

Ethernet II, Src: e2:fc:fe:95:25:5d (e2:fc:fe:95:25:5d), Dst: de:b0:ec:be:df:8a (de:b0:ec:be:df:8a)

Internet Protocol Version 4, Src: 10.0.0.2 (10.0.0.2), Dst: 10.0.0.1 (10.0.0.1)

Internet Control Message Protocol

No.	Time	Source	Destination	Protocol	Length	Info	Interface
0000	de b0 ec be df 8a e2 fc fe 95 25 5d 08 00 45 00%..E..					
0010	00 54 59 ea 40 00 40 01 cc bc 0a 00 00 02 0a 00	TY @ @					
0020	00 01 00 86 d5 a4 5c 66 00 01 70 26 88 5f 00 00	..z...f..p&....					
0030	00 00 7a 9b 04 00 00 00 00 00 10 11 12 13 14 15!#\$%					
0040	16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25!#\$%					
0050	26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35	&(')*,- ./012345					
0060	36 37	67					

Ready to load or capture

Packets: 135 · Displayed: 135 (100.0%)

Wireshark Screenshot 2 (Bottom):

Frame 31: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface s5-eth4, id 15

Ethernet II, Src: e2:fc:fe:95:25:5d (e2:fc:fe:95:25:5d), Dst: de:b0:ec:be:df:8a (de:b0:ec:be:df:8a)

Internet Protocol Version 4, Src: 10.0.0.2 (10.0.0.2), Dst: 10.0.0.1 (10.0.0.1)

Internet Control Message Protocol

No.	Time	Source	Destination	Protocol	Length	Info	Interface
12	0.0000076882	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 11)	s1-eth3
6	0.0000099406	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 5)	s1-eth1
31	0.001754694	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 32)	s5-eth4
2	0.001761306	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (no response found...)	s5-eth1
23	0.001762937	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (no response found...)	s2-eth3
28	0.001765483	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (no response found...)	s2-eth2
24	0.001766799	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 26)	s4-eth1
35	0.001768889	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 36)	s4-eth2
9	0.001770417	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 10)	s3-eth2
19	0.001772664	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 20)	s3-eth1
13	0.001773764	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 14)	s1-eth3
7	0.001775766	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) request id=0x5c66, seq=1/256, ttl=64 (reply in 8)	s1-eth1
8	0.001782622	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 7)	s1-eth1
14	0.001784077	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 13)	s1-eth3
20	0.001784534	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 19)	s3-eth1
10	0.001785632	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 9)	s3-eth2
36	0.001785965	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 35)	s4-eth2
26	0.001787226	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 24)	s4-eth3
16	0.001788600	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64	s5-eth2
32	0.001790128	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) reply id=0x5c66, seq=1/256, ttl=64 (request in 31)	s5-eth4
37	2.110848737	fe80::f829:80ff:fe...	ff02::2	ICMPv6	70	Router Solicitation from fa:29:80:a2:c9:6c	s3-eth3
38	2.110857244	fe80::f829:80ff:fe...	ff02::2	ICMPv6	70	Router Solicitation from fa:29:80:a2:c9:6c	s5-eth3

Ready to load or capture

Packets: 135 · Displayed: 135 (100.0%)

Here I use time to sort all of the traffic. And it shows the interface information here to show the traffic path. And from s1-s3-s4-s5 it shows the path A-C-E-D which is correct. And the interface flow is also correct from my code.

And then it flows from s5-s2-s4-s3-s1 which shows the path D-B-E-C-A. And the interface flow is also correct from my code.

- (g) OVS-OFCTL commands used to install the rules on switches (part 2). (If you use a controller, upload your controller program)

https://drive.google.com/drive/folders/1b_DUz_p68Y0iaSuvJY1R6mCJe--drP_e?usp=sharing

- (h) Also submit your custom topology files (.py files) for both topologies (part 2 & part 3) (**do NOT paste code in report**).

<https://drive.google.com/drive/folders/1psW28jz1h-h6xcpX9sr6mbfYU6AP2Ygv?usp=sharing>

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