Software-Defined Networking

Lab 4

Advanced Mininet, Open vSwitch, and SDN Controllers

University of Colorado Boulder

Department of Computer Science

Professor Levi Perigo, Ph.D.

Lab Summary

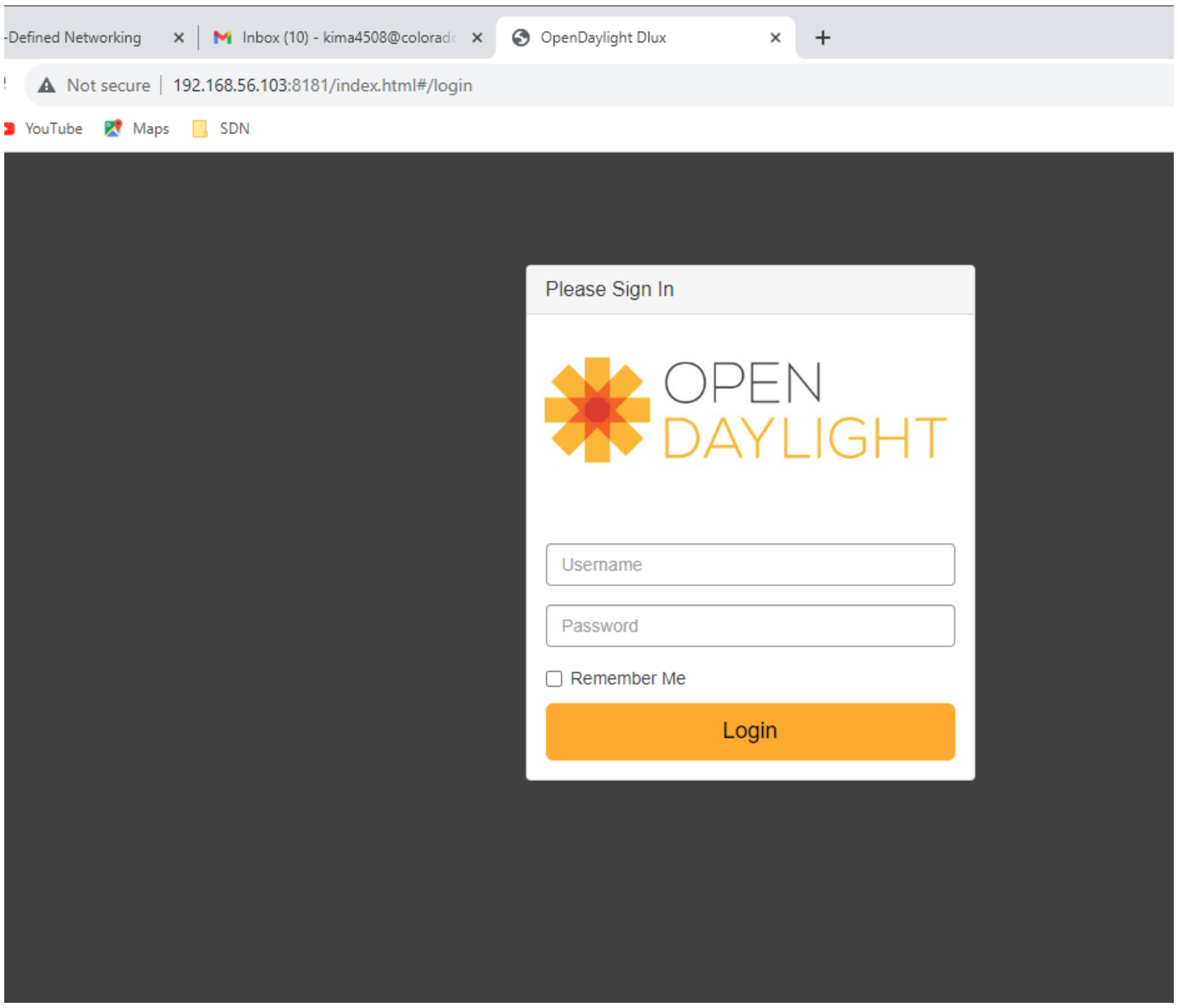
Understanding Mininet and popular SDN controllers is critical to understanding SDN. The purpose of this lab is to explore some of the advanced features of Mininet and become familiar with some of the popular SDN controllers in industry. Knowing how to use advanced features in Mininet and understanding specific commands and output, will aid in your understanding of SDN. The experience gained from using the controllers in this lab will facilitate your understanding of SDN controllers as you expand on the foundations from this lab in the rest of the course.

The objectives of this lab are to be used as guidelines, and additional exploration by the student is strongly encouraged.

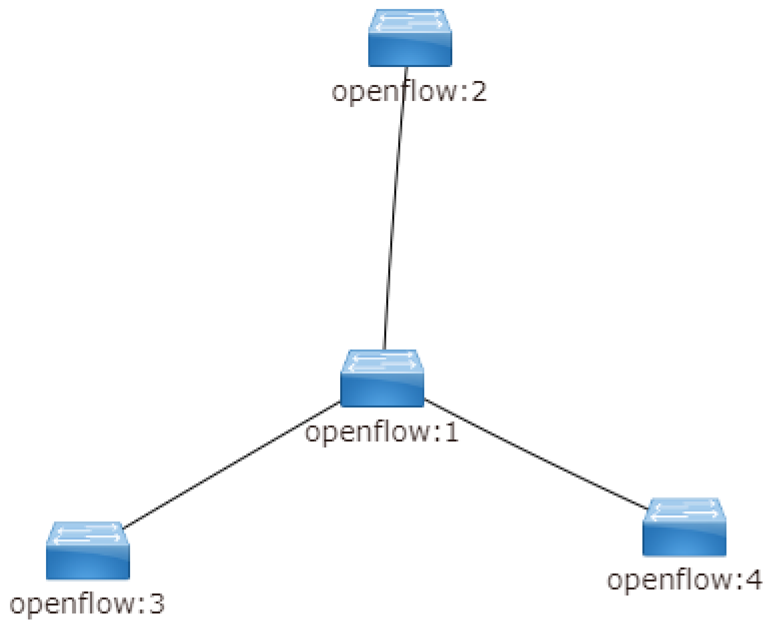
Objective 1:

PART A: Connect Mininet to OpenDaylight (ODL)

1. Refer the Lab 0 document to initialize OpenDaylight (ODL).
2. Enable the ODL web interface and all the features (use the Lab 0 document for assistance).
3. Login to the ODL web interface.



4. Login to Mininet.
    a. Start a new Mininet topology that connects to a remote controller (ODL), uses easy to read MAC addresses, uses OpenFlow v1.3, and uses a network topology that is a tree with depth of 2 and a fanout of 3.



    b. Explain in detail what the tree topology is including the depth and fanout settings. **[5 points]**

    The tree topology has two options:

    –depth=2 →

    The depth commands specify the number of levels beginning from the root.

    –fanout=3 →

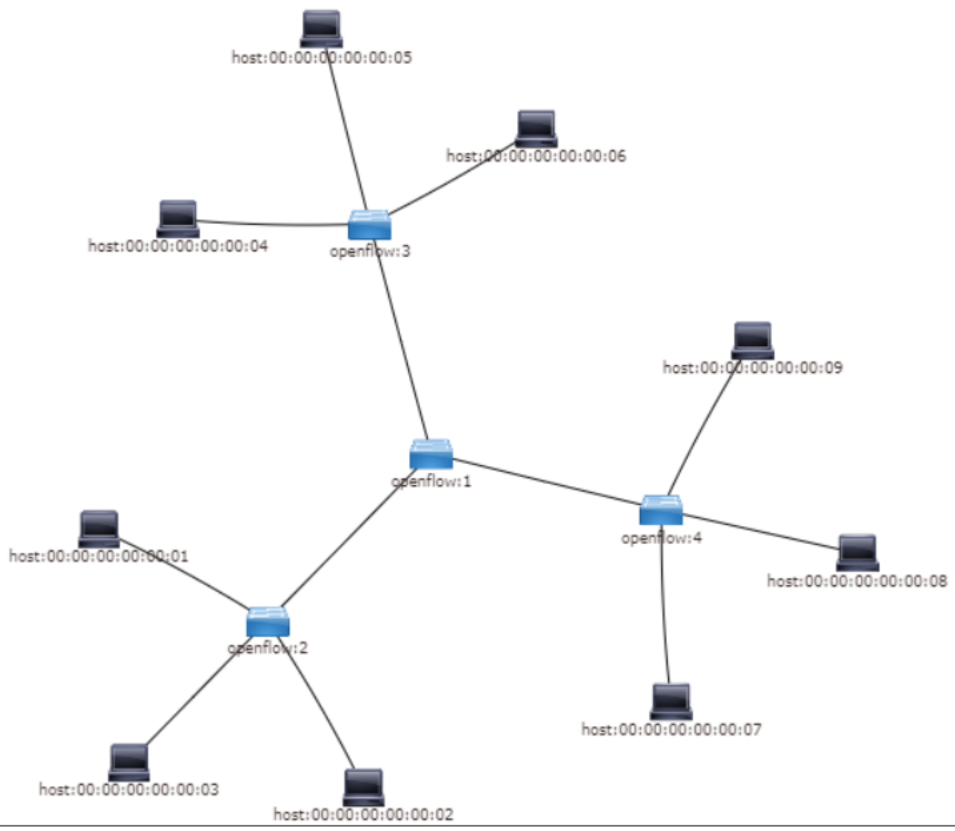
    The number of children each node has, is referred to as the fanout.

    In the topology above, the depth=2 and fanout=3 indicates that there are two levels and each node has 3 children.

5. Within the GUI of ODL, does the topology show any hosts connected to the switches? Why or why not?

No, because the switch doesn't have any request from the host and hence, there are no flows installed on the switch by the controller. The ODL controller would fetch flows from the switch to generate the diagram in the UI, but since there are no flows, the controller doesn't know about the hosts at all.

If not, how would you get the hosts in Mininet to be displayed in the ODL topology? **[5 points]**

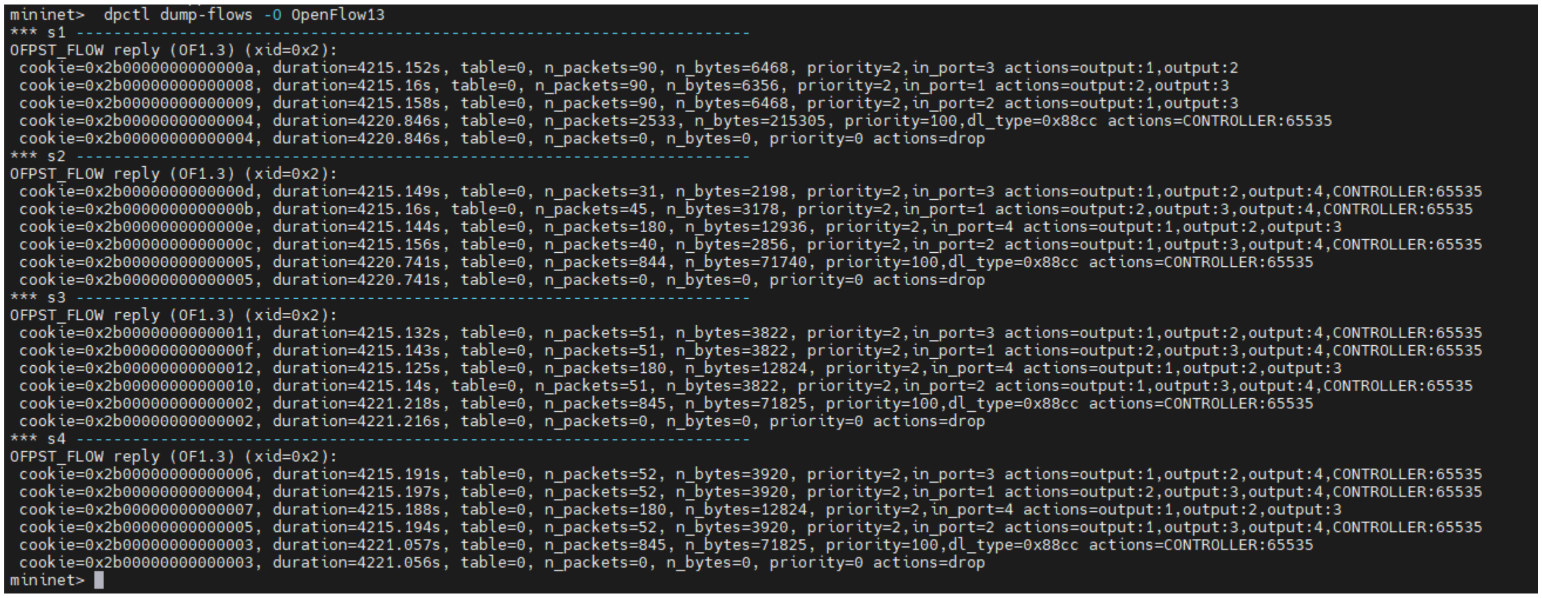


A simple ping between all hosts using "pingall" command would cause the controller to install appropriate flows and, when the GUI engine of the controller, fetches these flows, it would be able to display the topology.

6. Login to the Mininet VM with a duplicate session.

   a. Within the Mininet VM, you are going to issue Open vSwitch commands to view statistics and flow table entries of the switches.

n_packets, nbytes → represents the transmission units that caused flow hits (statistics)

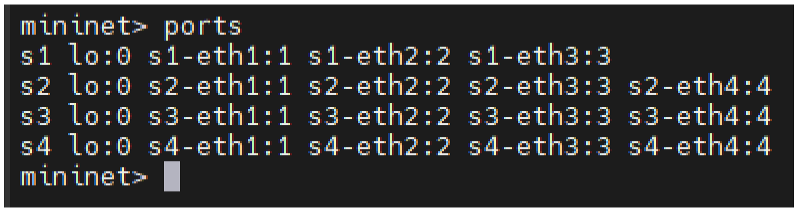


```
mininet>  dpctl dump-flows -O OpenFlow13
*** s1 ------------------------------------------------------------------------
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2b0000000000000a, duration=4215.152s, table=0, n_packets=90, n_bytes=6468, priority=2,in_port=3 actions=output:1,output:2
 cookie=0x2b00000000000008, duration=4215.16s, table=0, n_packets=90, n_bytes=6356, priority=2,in_port=1 actions=output:2,output:3
 cookie=0x2b00000000000009, duration=4215.158s, table=0, n_packets=90, n_bytes=6468, priority=2,in_port=2 actions=output:1,output:3
 cookie=0x2b00000000000004, duration=4220.846s, table=0, n_packets=2533, n_bytes=215305, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2b00000000000004, duration=4220.846s, table=0, n_packets=0, n_bytes=0, priority=0 actions=drop
*** s2 ------------------------------------------------------------------------
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2b0000000000000d, duration=4215.149s, table=0, n_packets=31, n_bytes=2198, priority=2,in_port=3 actions=output:1,output:2,output:4,CONTROLLER:65535
 cookie=0x2b0000000000000b, duration=4215.16s, table=0, n_packets=45, n_bytes=3178, priority=2,in_port=1 actions=output:2,output:3,output:4,CONTROLLER:65535
 cookie=0x2b0000000000000e, duration=4215.144s, table=0, n_packets=180, n_bytes=12936, priority=2,in_port=4 actions=output:1,output:2,output:3
 cookie=0x2b0000000000000c, duration=4215.156s, table=0, n_packets=40, n_bytes=2856, priority=2,in_port=2 actions=output:1,output:3,output:4,CONTROLLER:65535
 cookie=0x2b00000000000005, duration=4220.741s, table=0, n_packets=844, n_bytes=71740, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2b00000000000005, duration=4220.741s, table=0, n_packets=0, n_bytes=0, priority=0 actions=drop
*** s3 ------------------------------------------------------------------------
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2b00000000000011, duration=4215.132s, table=0, n_packets=51, n_bytes=3822, priority=2,in_port=3 actions=output:1,output:2,output:4,CONTROLLER:65535
 cookie=0x2b0000000000000f, duration=4215.143s, table=0, n_packets=51, n_bytes=3822, priority=2,in_port=1 actions=output:2,output:3,output:4,CONTROLLER:65535
 cookie=0x2b00000000000012, duration=4215.125s, table=0, n_packets=180, n_bytes=12824, priority=2,in_port=4 actions=output:1,output:2,output:3
 cookie=0x2b00000000000010, duration=4215.14s, table=0, n_packets=51, n_bytes=3822, priority=2,in_port=2 actions=output:1,output:3,output:4,CONTROLLER:65535
 cookie=0x2b00000000000002, duration=4221.218s, table=0, n_packets=845, n_bytes=71825, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2b00000000000002, duration=4221.216s, table=0, n_packets=0, n_bytes=0, priority=0 actions=drop
*** s4 ------------------------------------------------------------------------
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2b00000000000006, duration=4215.191s, table=0, n_packets=52, n_bytes=3920, priority=2,in_port=3 actions=output:1,output:2,output:4,CONTROLLER:65535
 cookie=0x2b00000000000004, duration=4215.197s, table=0, n_packets=52, n_bytes=3920, priority=2,in_port=1 actions=output:2,output:3,output:4,CONTROLLER:65535
 cookie=0x2b00000000000007, duration=4215.188s, table=0, n_packets=180, n_bytes=12824, priority=2,in_port=4 actions=output:1,output:2,output:3
 cookie=0x2b00000000000005, duration=4215.194s, table=0, n_packets=52, n_bytes=3920, priority=2,in_port=2 actions=output:1,output:3,output:4,CONTROLLER:65535
 cookie=0x2b00000000000003, duration=4221.057s, table=0, n_packets=845, n_bytes=71825, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2b00000000000003, duration=4221.056s, table=0, n_packets=0, n_bytes=0, priority=0 actions=drop
mininet>
```

b. Provide a combined screenshot for the following operations: **[5 points]**

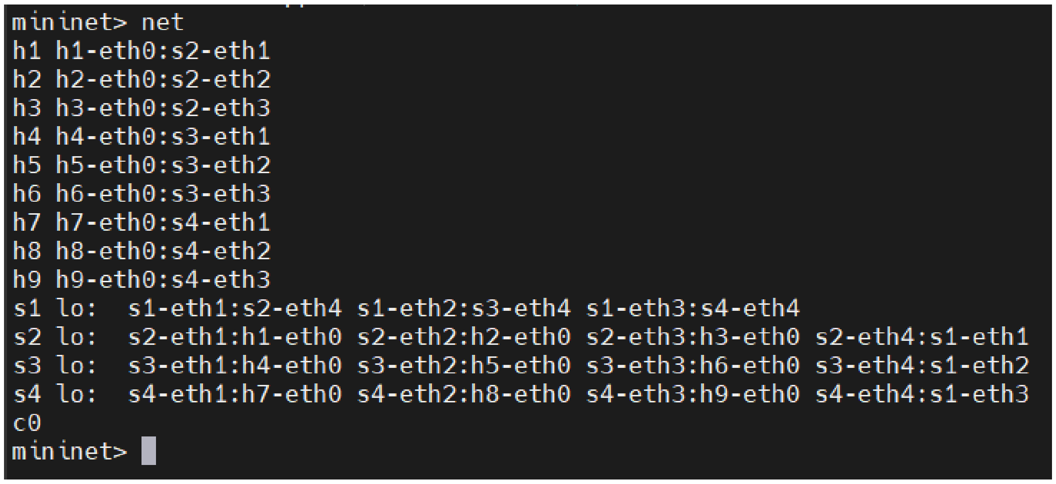
i. Command to show all ports and connections for the network.

Ports -



```
mininet> ports
s1 lo:0 s1-eth1:1 s1-eth2:2 s1-eth3:3
s2 lo:0 s2-eth1:1 s2-eth2:2 s2-eth3:3 s2-eth4:4
s3 lo:0 s3-eth1:1 s3-eth2:2 s3-eth3:3 s3-eth4:4
s4 lo:0 s4-eth1:1 s4-eth2:2 s4-eth3:3 s4-eth4:4
mininet>
```

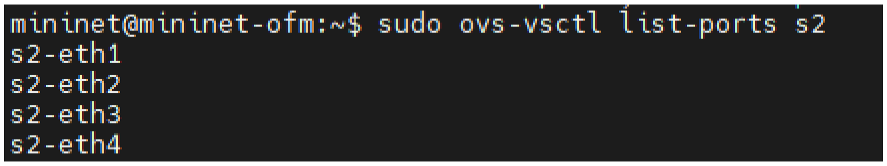
Connections -



```
mininet> net
h1 h1-eth0:s2-eth1
h2 h2-eth0:s2-eth2
h3 h3-eth0:s2-eth3
h4 h4-eth0:s3-eth1
h5 h5-eth0:s3-eth2
h6 h6-eth0:s3-eth3
h7 h7-eth0:s4-eth1
h8 h8-eth0:s4-eth2
h9 h9-eth0:s4-eth3
s1 lo:  s1-eth1:s2-eth4 s1-eth2:s3-eth4 s1-eth3:s4-eth4
s2 lo:  s2-eth1:h1-eth0 s2-eth2:h2-eth0 s2-eth3:h3-eth0 s2-eth4:s1-eth1
s3 lo:  s3-eth1:h4-eth0 s3-eth2:h5-eth0 s3-eth3:h6-eth0 s3-eth4:s1-eth2
s4 lo:  s4-eth1:h7-eth0 s4-eth2:h8-eth0 s4-eth3:h9-eth0 s4-eth4:s1-eth3
c0
mininet>
```

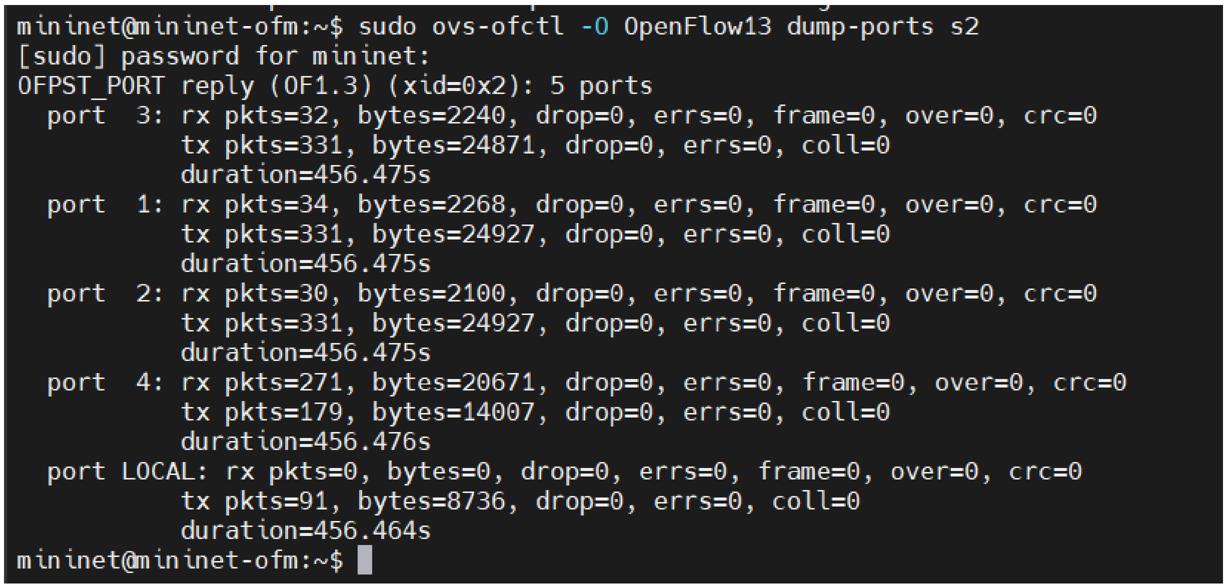
ii. Command to show all ports for Switch 2.

Command to see all ports from mininet perspective



```
mininet@mininet-ofm:~$ sudo ovs-vsctl list-ports s2
s2-eth1
s2-eth2
s2-eth3
s2-eth4
```

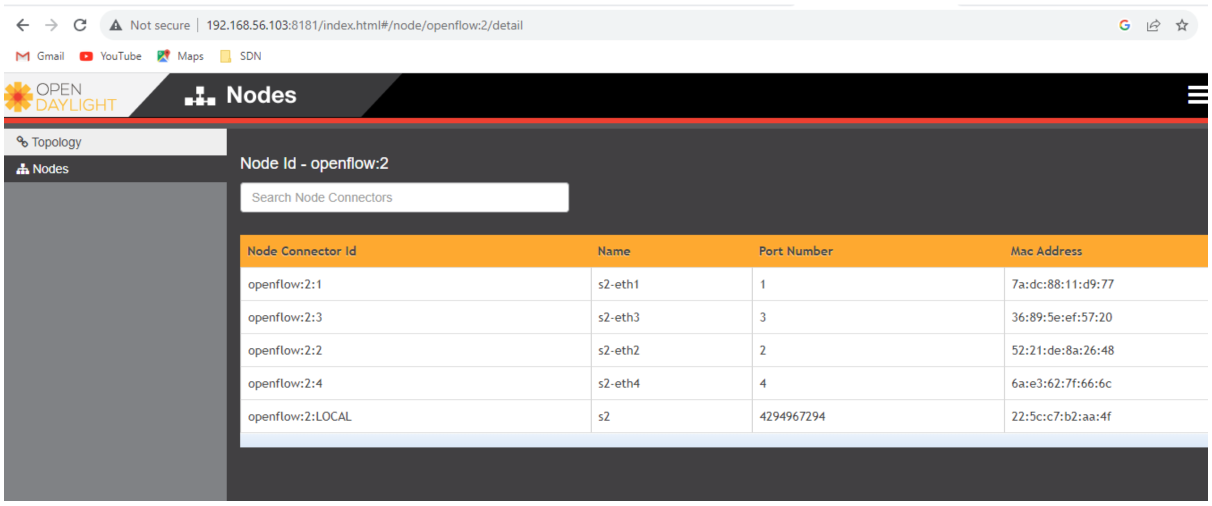
command to see all ports, to verify with the flow entries



```
mininet@mininet-ofm:~$ sudo ovs-ofctl -O OpenFlow13 dump-ports s2
[sudo] password for mininet:
OFPST_PORT reply (OF1.3) (xid=0x2): 5 ports
  port  3: rx pkts=32, bytes=2240, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=331, bytes=24871, drop=0, errs=0, coll=0
           duration=456.475s
  port  1: rx pkts=34, bytes=2268, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=331, bytes=24927, drop=0, errs=0, coll=0
           duration=456.475s
  port  2: rx pkts=30, bytes=2100, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=331, bytes=24927, drop=0, errs=0, coll=0
           duration=456.475s
  port  4: rx pkts=271, bytes=20671, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=179, bytes=14007, drop=0, errs=0, coll=0
           duration=456.476s
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=91, bytes=8736, drop=0, errs=0, coll=0
           duration=456.464s
mininet@mininet-ofm:~$
```

c. Write steps or provide a screenshot of the same functionality as in [b (ii)] in the GUI of ODL. **[5 points]**

karaf version 0.7.0 -> contains odl-dlux-nodes



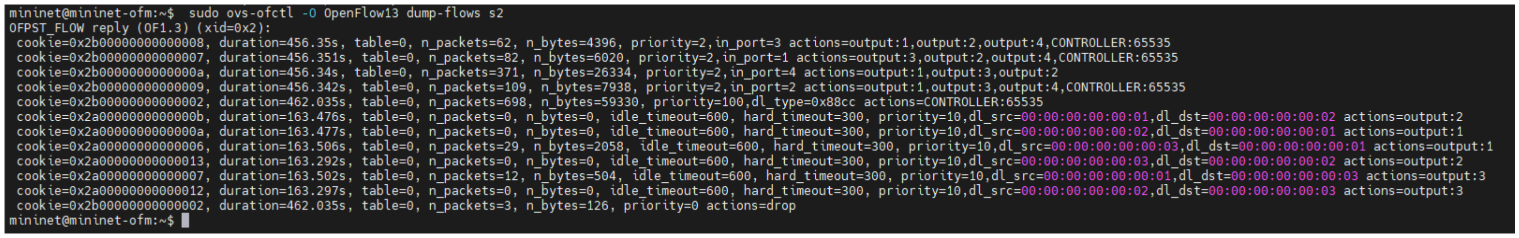
Node Id - openflow:2

Node Connector Id	Name	Port Number	Mac Address
openflow:2:1	s2-eth1	1	7a:dc:88:11:d9:77
openflow:2:3	s2-eth3	3	36:89:5e:ef:57:20
openflow:2:2	s2-eth2	2	52:21:de:8a:26:48
openflow:2:4	s2-eth4	4	6a:e3:62:7f:66:6c
openflow:2:LOCAL	s2	4294967294	22:5c:c7:b2:aa:4f

d. Issue the command that shows all the current flow table flow entries in switch 2.

i. What command did you use? **[5 points]**

sudo ovs-ofctl -O OpenFlow13 dump-flows s2

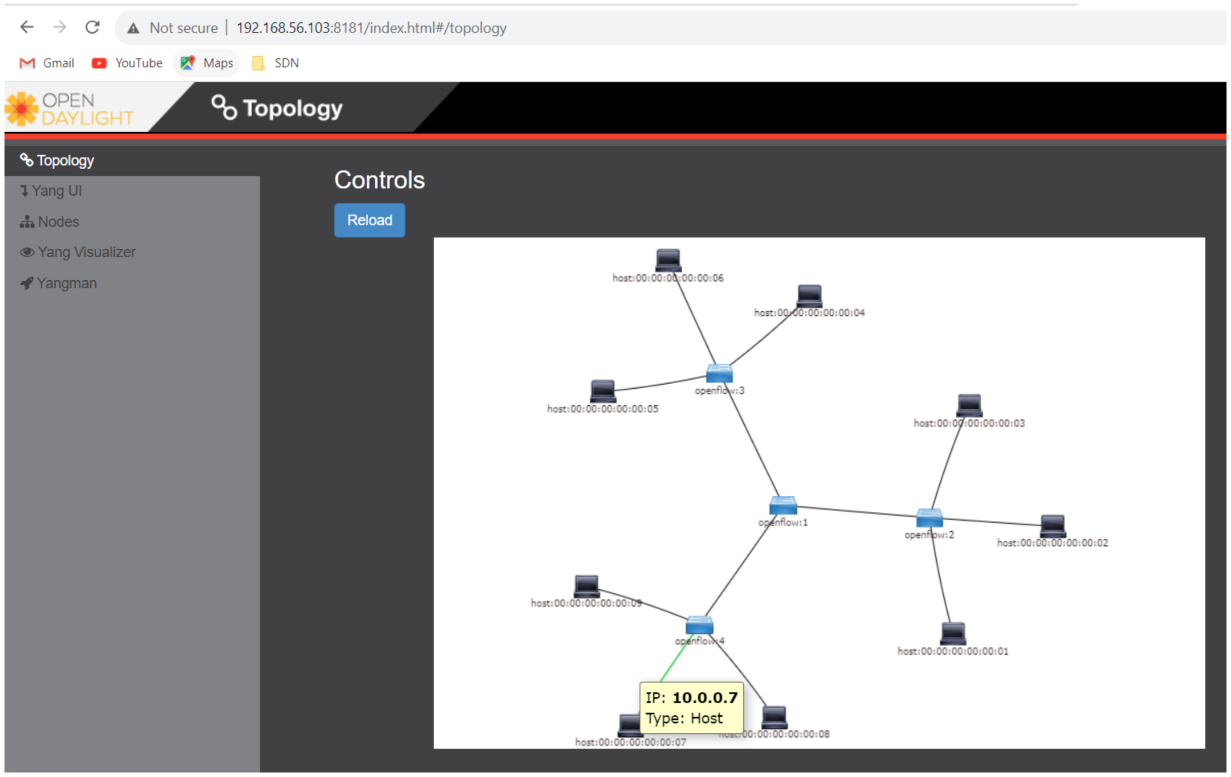


```
mininet@mininet-ofm:~$  sudo ovs-ofctl -O OpenFlow13 dump-flows s2
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2b00000000000008, duration=456.35s, table=0, n_packets=62, n_bytes=4396, priority=2,in_port=3 actions=output:1,output:2,output:4,CONTROLLER:65535
 cookie=0x2b00000000000007, duration=456.351s, table=0, n_packets=82, n_bytes=6020, priority=2,in_port=1 actions=output:3,output:2,output:4,CONTROLLER:65535
 cookie=0x2b0000000000000a, duration=456.34s, table=0, n_packets=371, n_bytes=26334, priority=2,in_port=4 actions=output:1,output:3,output:2
 cookie=0x2b00000000000009, duration=456.342s, table=0, n_packets=109, n_bytes=7938, priority=2,in_port=2 actions=output:1,output:3,output:4,CONTROLLER:65535
 cookie=0x2b00000000000002, duration=462.035s, table=0, n_packets=698, n_bytes=59330, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2a0000000000000b, duration=163.476s, table=0, n_packets=0, n_bytes=0, idle_timeout=600, hard_timeout=300, priority=10,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x2a0000000000000a, duration=163.477s, table=0, n_packets=0, n_bytes=0, idle_timeout=600, hard_timeout=300, priority=10,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01 actions=output:1
 cookie=0x2a00000000000006, duration=163.506s, table=0, n_packets=29, n_bytes=2058, idle_timeout=600, hard_timeout=300, priority=10,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:01 actions=output:1
 cookie=0x2a00000000000013, duration=163.292s, table=0, n_packets=0, n_bytes=0, idle_timeout=600, hard_timeout=300, priority=10,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x2a00000000000007, duration=163.502s, table=0, n_packets=12, n_bytes=504, idle_timeout=600, hard_timeout=300, priority=10,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03 actions=output:3
 cookie=0x2a00000000000012, duration=163.297s, table=0, n_packets=0, n_bytes=0, idle_timeout=600, hard_timeout=300, priority=10,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03 actions=output:3
 cookie=0x2b00000000000002, duration=462.035s, table=0, n_packets=3, n_bytes=126, priority=0 actions=drop
mininet@mininet-ofm:~$ 
```

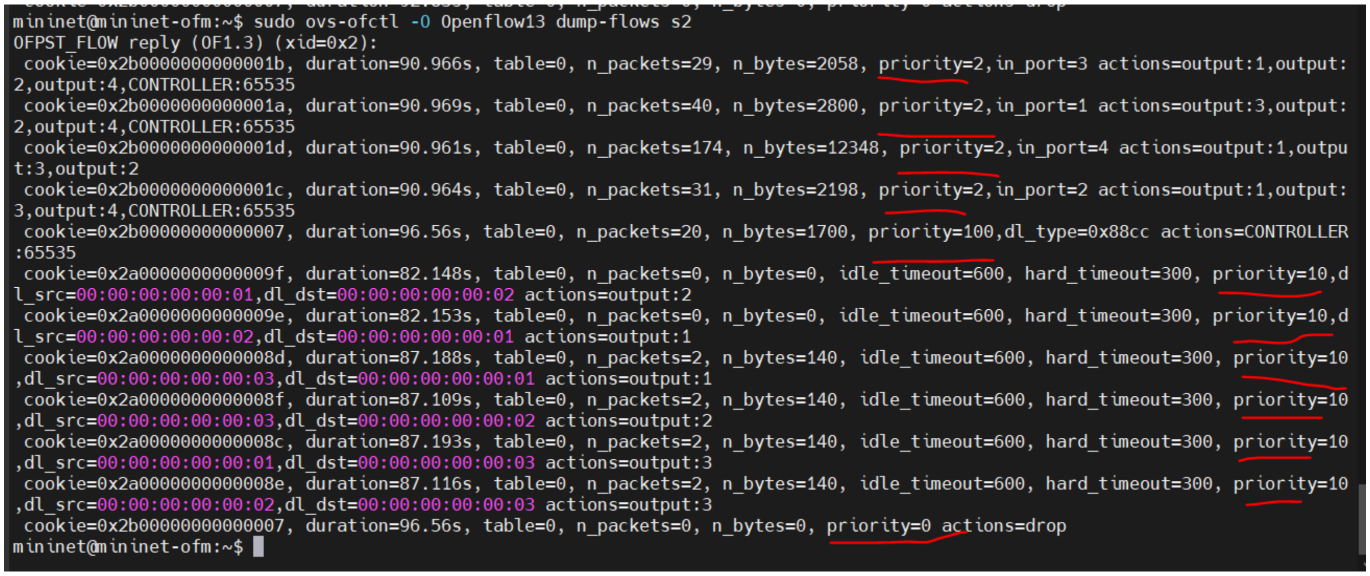
ii. Provide a screenshot of this in the GUI of ODL**. [5 points]**

—> Flows are not visible in ODL 0.7.0, possible bug
—> 0.8.0 Dlux feature is deprecated and is not present.

After Pingall >>



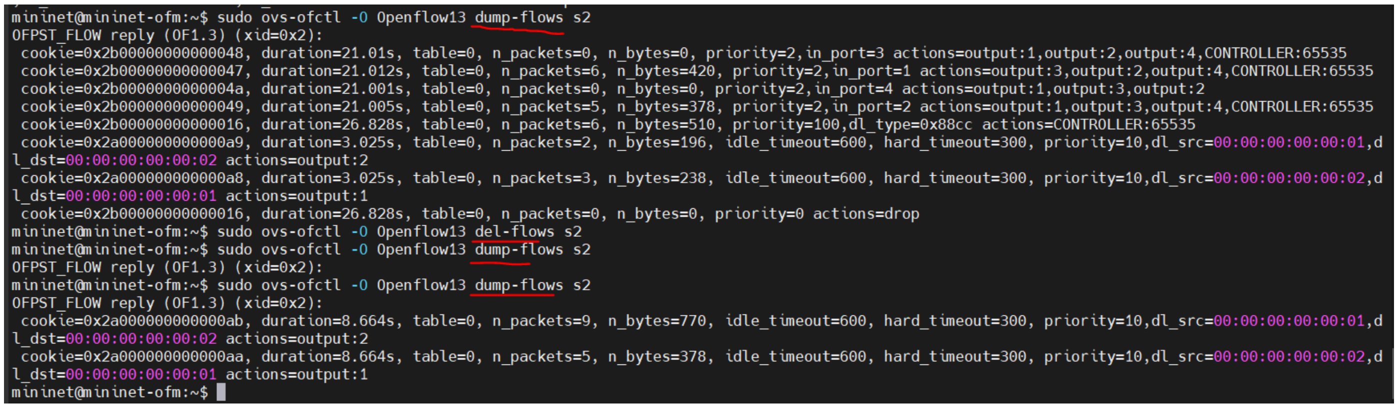
iii. What is the flow entry with the lowest priority value? Explain what that entry is. **[5 points]**



```
mininet@mininet-ofm:~$ sudo ovs-ofctl -O Openflow13 dump-flows s2
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2b0000000000001b, duration=90.966s, table=0, n_packets=29, n_bytes=2058, priority=2,in_port=3 actions=output:1,output:
2,output:4,CONTROLLER:65535
 cookie=0x2b0000000000001a, duration=90.969s, table=0, n_packets=40, n_bytes=2800, priority=2,in_port=1 actions=output:3,output:
2,output:4,CONTROLLER:65535
 cookie=0x2b0000000000001d, duration=90.961s, table=0, n_packets=174, n_bytes=12348, priority=2,in_port=4 actions=output:1,outpu
t:3,output:2
 cookie=0x2b0000000000001c, duration=90.964s, table=0, n_packets=31, n_bytes=2198, priority=2,in_port=2 actions=output:1,output:
3,output:4,CONTROLLER:65535
 cookie=0x2b00000000000007, duration=96.56s, table=0, n_packets=20, n_bytes=1700, priority=100,dl_type=0x88cc actions=CONTROLLER
:65535
 cookie=0x2a0000000000009f, duration=82.148s, table=0, n_packets=0, n_bytes=0, idle_timeout=600, hard_timeout=300, priority=10,d
l_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x2a0000000000009e, duration=82.153s, table=0, n_packets=0, n_bytes=0, idle_timeout=600, hard_timeout=300, priority=10,d
l_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01 actions=output:1
 cookie=0x2a0000000000008d, duration=87.188s, table=0, n_packets=2, n_bytes=140, idle_timeout=600, hard_timeout=300, priority=10
,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:01 actions=output:1
 cookie=0x2a0000000000008f, duration=87.109s, table=0, n_packets=2, n_bytes=140, idle_timeout=600, hard_timeout=300, priority=10
,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x2a0000000000008c, duration=87.193s, table=0, n_packets=2, n_bytes=140, idle_timeout=600, hard_timeout=300, priority=10
,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03 actions=output:3
 cookie=0x2a0000000000008e, duration=87.116s, table=0, n_packets=2, n_bytes=140, idle_timeout=600, hard_timeout=300, priority=10
,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03 actions=output:3
 cookie=0x2b00000000000007, duration=96.56s, table=0, n_packets=0, n_bytes=0, priority=0 actions=drop
mininet@mininet-ofm:~$
```

Priority 0 - the **last flow matches anything**, it basically drops anything if there are no flows or if there is no information about flows from the controller. It is like the last decision, which will be taken when no flows are matched.

iv. Generate traffic from hosts connected to switch 2 and refresh the flow table; were additional flow entries added? Why or why not? **[5 points]**



```
mininet@mininet-ofm:~$ sudo ovs-ofctl -O Openflow13 dump-flows s2
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2b00000000000048, duration=21.01s, table=0, n_packets=0, n_bytes=0, priority=2,in_port=3 actions=output:1,output:2,output:4,CONTROLLER:65535
 cookie=0x2b00000000000047, duration=21.012s, table=0, n_packets=6, n_bytes=420, priority=2,in_port=1 actions=output:3,output:2,output:4,CONTROLLER:65535
 cookie=0x2b0000000000004a, duration=21.001s, table=0, n_packets=0, n_bytes=0, priority=2,in_port=4 actions=output:1,output:3,output:2
 cookie=0x2b00000000000049, duration=21.005s, table=0, n_packets=5, n_bytes=378, priority=2,in_port=2 actions=output:1,output:3,output:4,CONTROLLER:65535
 cookie=0x2b00000000000016, duration=26.828s, table=0, n_packets=6, n_bytes=510, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2a000000000000a9, duration=3.025s, table=0, n_packets=2, n_bytes=196, idle_timeout=600, hard_timeout=300, priority=10,dl_src=00:00:00:00:00:01,d
l_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x2a000000000000a8, duration=3.025s, table=0, n_packets=3, n_bytes=238, idle_timeout=600, hard_timeout=300, priority=10,dl_src=00:00:00:00:00:02,d
l_dst=00:00:00:00:00:01 actions=output:1
 cookie=0x2b00000000000016, duration=26.828s, table=0, n_packets=0, n_bytes=0, priority=0 actions=drop
mininet@mininet-ofm:~$ sudo ovs-ofctl -O Openflow13 del-flows s2
mininet@mininet-ofm:~$ sudo ovs-ofctl -O Openflow13 dump-flows s2
OFPST_FLOW reply (OF1.3) (xid=0x2):
mininet@mininet-ofm:~$ sudo ovs-ofctl -O Openflow13 dump-flows s2
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2a000000000000ab, duration=8.664s, table=0, n_packets=9, n_bytes=770, idle_timeout=600, hard_timeout=300, priority=10,dl_src=00:00:00:00:00:01,d
l_dst=00:00:00:00:00:02 actions=output:2
 cookie=0x2a000000000000aa, duration=8.664s, table=0, n_packets=5, n_bytes=378, idle_timeout=600, hard_timeout=300, priority=10,dl_src=00:00:00:00:00:02,d
l_dst=00:00:00:00:00:01 actions=output:1
mininet@mininet-ofm:~$
```

The flows are not installed if they are flushed from the switch, since the controller doesn't maintain a record of the flows on each switch. **In the above snip**, there are no flows printed after the flows are deleted. Only when a new request comes in on a port, punted by the switch to the controller , the controller would install that specific flow. **This can be observed in the second dump-flows statement.**

The controller itself would not maintain these flows but only have holistic view and decision making ability.

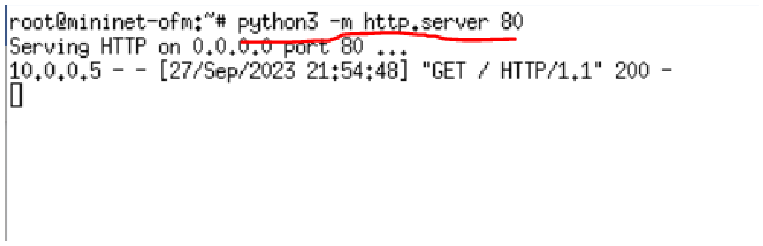
e. Find HTTP Flows

i. On Host1 run a HTTP server.

ii. Perform a “wget” from H5 to the web server (H1).

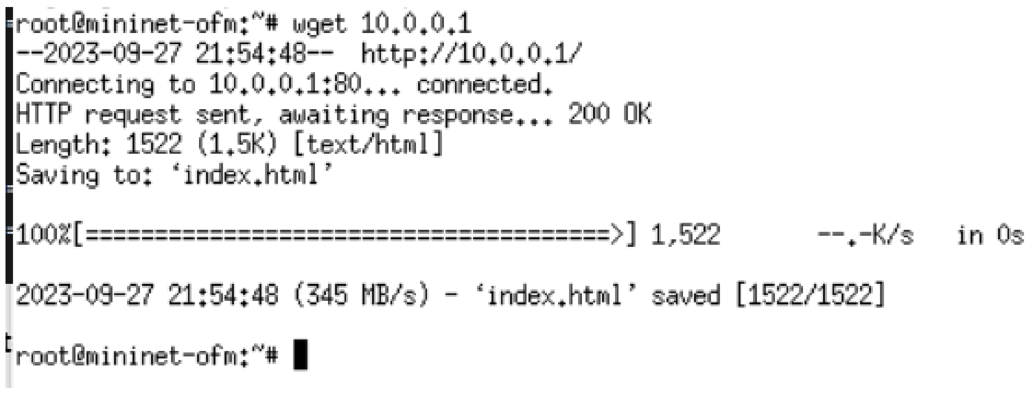
1. What command did you use? **[5 points]**

h1 terminal



```
root@mininet-ofm:~# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 ...
10.0.0.5 - - [27/Sep/2023 21:54:48] "GET / HTTP/1.1" 200 -
```

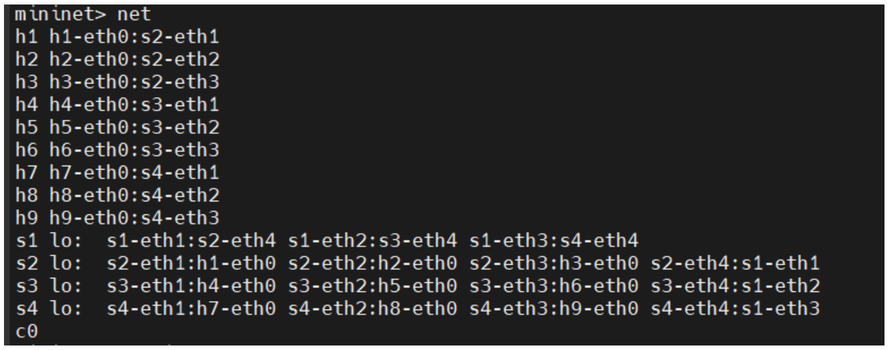
h5 terminal



```
root@mininet-ofm:~# wget 10.0.0.1
--2023-09-27 21:54:48--  http://10.0.0.1/
Connecting to 10.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1522 (1.5K) [text/html]
Saving to: ‘index.html’

100%[=====================================>] 1,522       --.-K/s    in 0s

2023-09-27 21:54:48 (345 MB/s) - ‘index.html’ saved [1522/1522]

root@mininet-ofm:~#
```

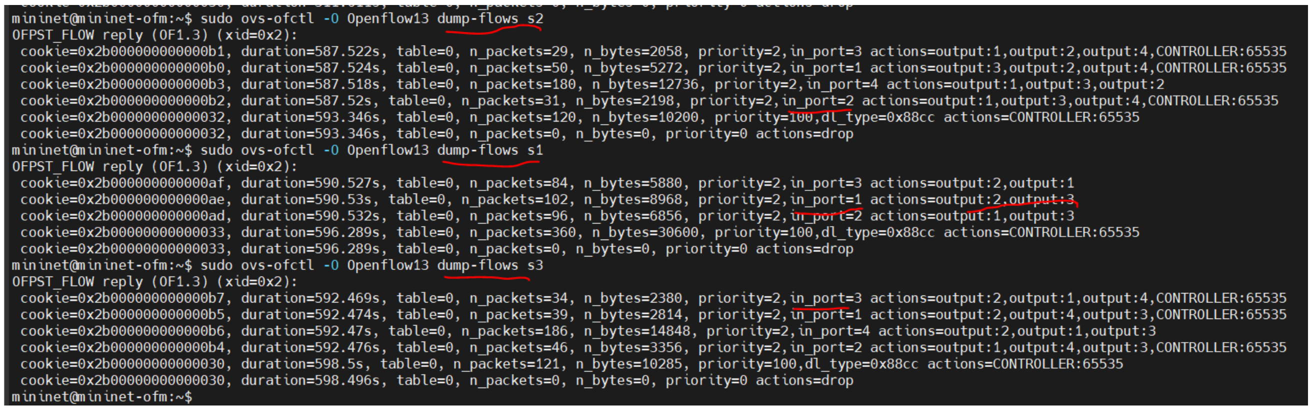
iii. Within ODL GUI or OVS, select a switch in the path, and highlight the flow entry that shows the HTTP traffic **[10 points]**

The http traffic is not necessarily a match with http port and tcp protocol, since the topology isn’t that complicated and the controller itself would usually install Ethernet header based flows, since the source and destination hosts are in the same subnet.



```
mininet> net
h1 h1-eth0:s2-eth1
h2 h2-eth0:s2-eth2
h3 h3-eth0:s2-eth3
h4 h4-eth0:s3-eth1
h5 h5-eth0:s3-eth2
h6 h6-eth0:s3-eth3
h7 h7-eth0:s4-eth1
h8 h8-eth0:s4-eth2
h9 h9-eth0:s4-eth3
s1 lo:  s1-eth1:s2-eth4 s1-eth2:s3-eth4 s1-eth3:s4-eth4
s2 lo:  s2-eth1:h1-eth0 s2-eth2:h2-eth0 s2-eth3:h3-eth0 s2-eth4:s1-eth1
s3 lo:  s3-eth1:h4-eth0 s3-eth2:h5-eth0 s3-eth3:h6-eth0 s3-eth4:s1-eth2
s4 lo:  s4-eth1:h7-eth0 s4-eth2:h8-eth0 s4-eth3:h9-eth0 s4-eth4:s1-eth3
c0
```

Connections : h1 < - - - > s2 < - - - > s1 < - - - - > s3 < - - - - > h5



```
mininet@mininet-ofm:~$ sudo ovs-ofctl -O Openflow13 dump-flows s2
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2b000000000000b1, duration=587.522s, table=0, n_packets=29, n_bytes=2058, priority=2,in_port=3 actions=output:1,output:2,output:4,CONTROLLER:65535
 cookie=0x2b000000000000b0, duration=587.524s, table=0, n_packets=50, n_bytes=5272, priority=2,in_port=1 actions=output:3,output:2,output:4,CONTROLLER:65535
 cookie=0x2b000000000000b3, duration=587.518s, table=0, n_packets=180, n_bytes=12736, priority=2,in_port=4 actions=output:1,output:3,output:2
 cookie=0x2b000000000000b2, duration=587.52s, table=0, n_packets=31, n_bytes=2198, priority=2,in_port=2 actions=output:1,output:3,output:4,CONTROLLER:65535
 cookie=0x2b00000000000032, duration=593.346s, table=0, n_packets=120, n_bytes=10200, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2b00000000000032, duration=593.346s, table=0, n_packets=0, n_bytes=0, priority=0 actions=drop
mininet@mininet-ofm:~$ sudo ovs-ofctl -O Openflow13 dump-flows s1
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2b000000000000af, duration=590.527s, table=0, n_packets=84, n_bytes=5880, priority=2,in_port=3 actions=output:2,output:1
 cookie=0x2b000000000000ae, duration=590.53s, table=0, n_packets=102, n_bytes=8968, priority=2,in_port=1 actions=output:2,output:3
 cookie=0x2b000000000000ad, duration=590.532s, table=0, n_packets=96, n_bytes=6856, priority=2,in_port=2 actions=output:1,output:3
 cookie=0x2b00000000000033, duration=596.289s, table=0, n_packets=360, n_bytes=30600, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2b00000000000033, duration=596.289s, table=0, n_packets=0, n_bytes=0, priority=0 actions=drop
mininet@mininet-ofm:~$ sudo ovs-ofctl -O Openflow13 dump-flows s3
OFPST_FLOW reply (OF1.3) (xid=0x2):
 cookie=0x2b000000000000b7, duration=592.469s, table=0, n_packets=34, n_bytes=2380, priority=2,in_port=3 actions=output:2,output:1,output:4,CONTROLLER:65535
 cookie=0x2b000000000000b5, duration=592.474s, table=0, n_packets=39, n_bytes=2814, priority=2,in_port=1 actions=output:2,output:4,output:3,CONTROLLER:65535
 cookie=0x2b000000000000b6, duration=592.47s, table=0, n_packets=186, n_bytes=14848, priority=2,in_port=4 actions=output:2,output:1,output:3
 cookie=0x2b000000000000b4, duration=592.476s, table=0, n_packets=46, n_bytes=3356, priority=2,in_port=2 actions=output:1,output:4,output:3,CONTROLLER:65535
 cookie=0x2b00000000000030, duration=598.5s, table=0, n_packets=121, n_bytes=10285, priority=100,dl_type=0x88cc actions=CONTROLLER:65535
 cookie=0x2b00000000000030, duration=598.496s, table=0, n_packets=0, n_bytes=0, priority=0 actions=drop
mininet@mininet-ofm:~$
```

Each flow as observed above is flooding the packets on all the ports except the source port on each switch. This would cause the http packet to reach the destination eventually. Although this doesn't seem to be the correct way, better applications on top of the controller can install more appropriate flows.

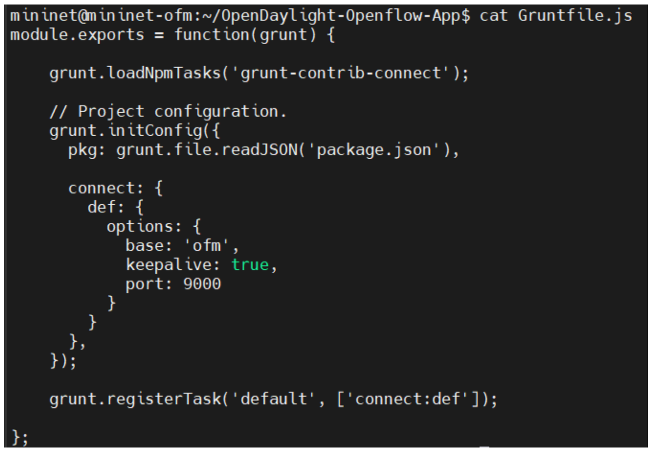
f. Exit and clean up Mininet.

PART B: Flow management with Cisco OpenFlow Manager (OFM)

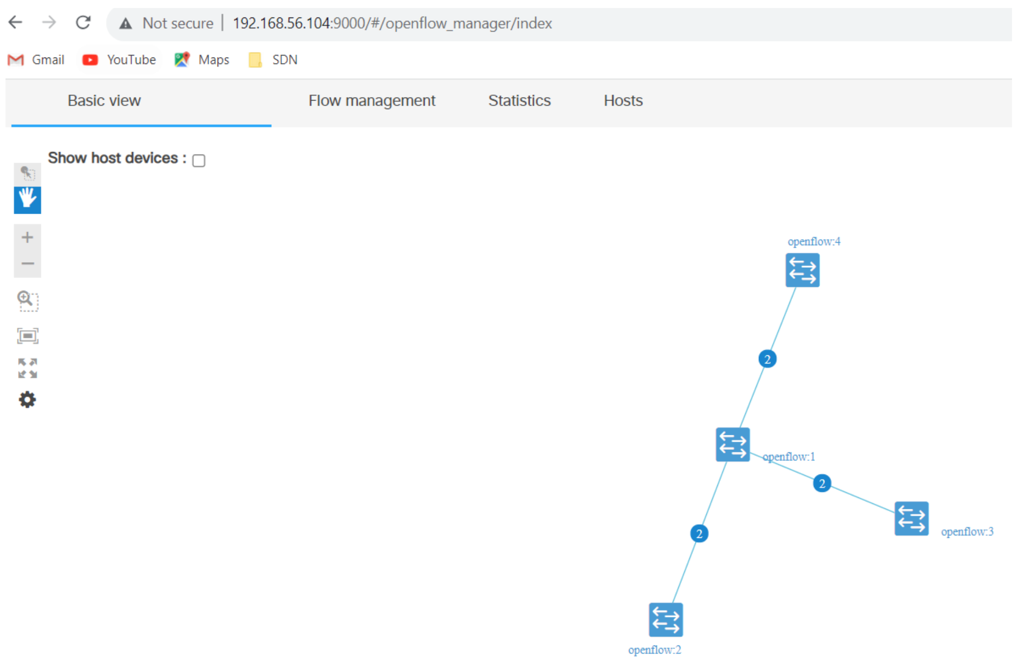
1. Refer the Lab 0 document to configure and initialize OFM.
2. Explain what the grunt command does? **[5 points]**

grunt command helps to perform repetitive tasks.

For OFM it starts up the webserver on port 9000.



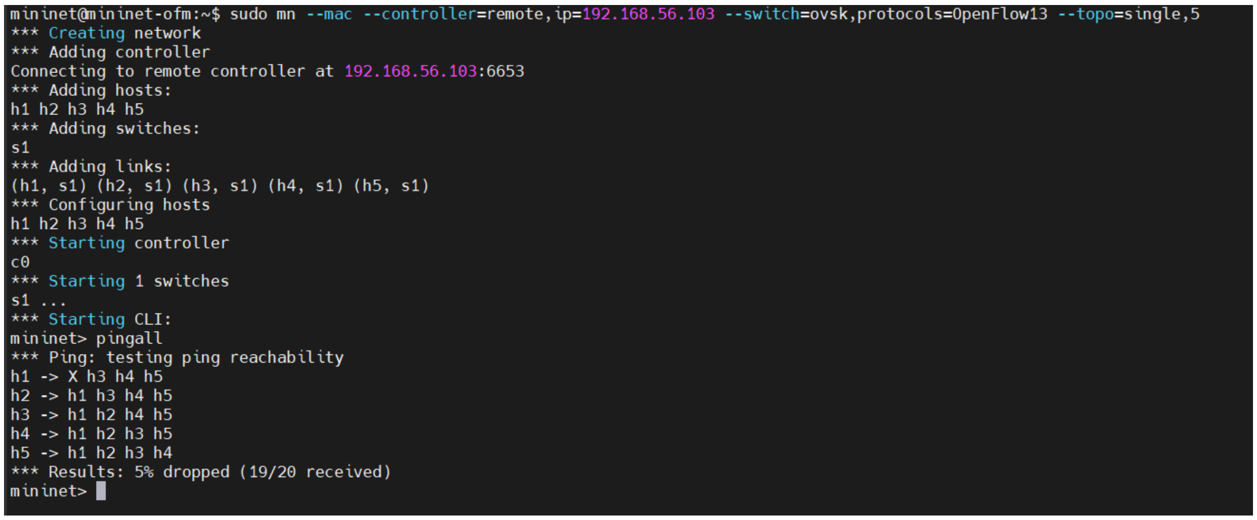
```
mininet@mininet-ofm:~/OpenDaylight-Openflow-App$ cat Gruntfile.js
module.exports = function(grunt) {

    grunt.loadNpmTasks('grunt-contrib-connect');

    // Project configuration.
    grunt.initConfig({
      pkg: grunt.file.readJSON('package.json'),

      connect: {
        def: {
          options: {
            base: 'ofm',
            keepalive: true,
            port: 9000
          }
        }
      },
    });

    grunt.registerTask('default', ['connect:def']);

};
```



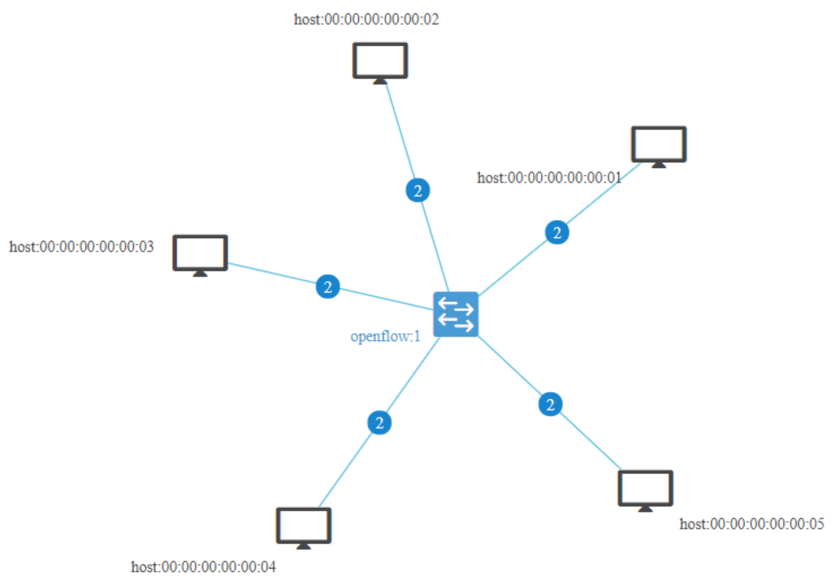
3. On the ODL VM, start the ODL controller by installing the following features:

   feature:install odl-restconf-all odl-openflowplugin-all odl-l2switch-all webconsole

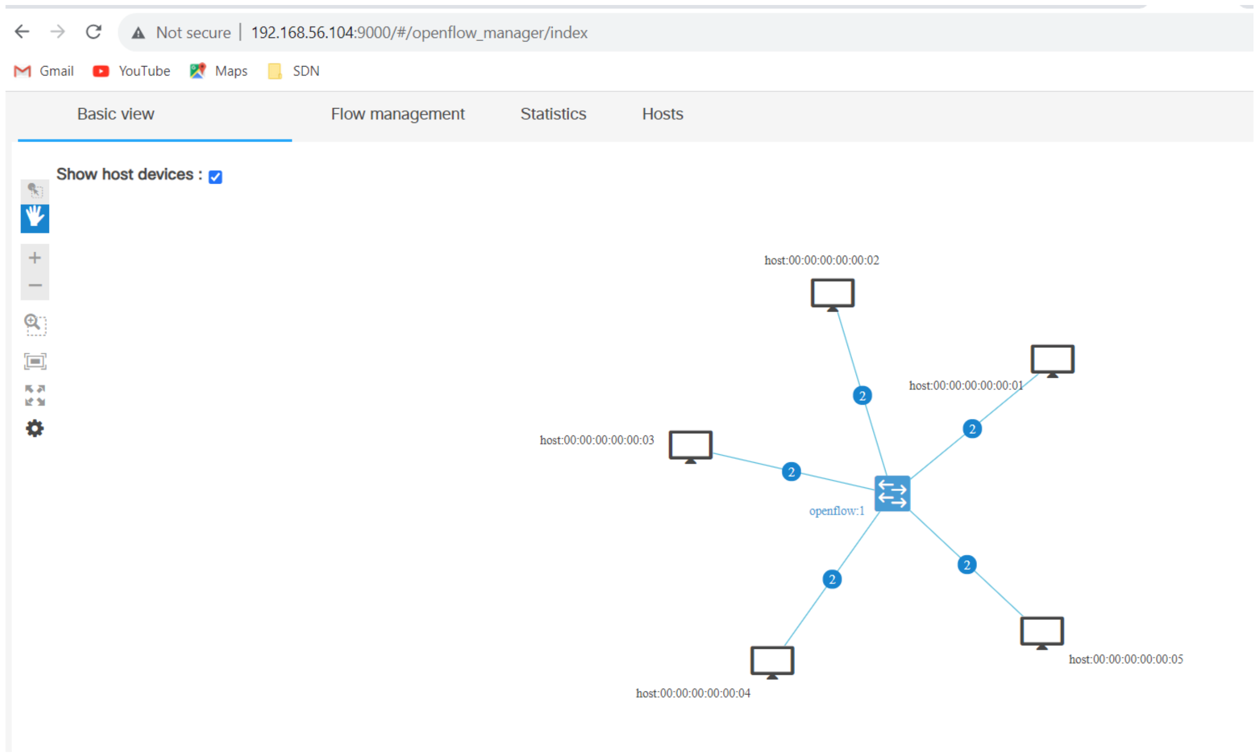
4. On Mininet, clear any existing Mininet topologies and build a new topology consisting of a single switch and five hosts, easy to read mac addresses, ODL as remote controller, running on OpenFlow 1.3 protocol. Paste the output of the pingall command. **[5 points]**.



```
mininet@mininet-ofm:~$ sudo mn --mac --controller=remote,ip=192.168.56.103 --switch=ovsk,protocols=OpenFlow13 --topo=single,5
*** Creating network
*** Adding controller
Connecting to remote controller at 192.168.56.103:6653
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 5% dropped (19/20 received)
mininet>
```

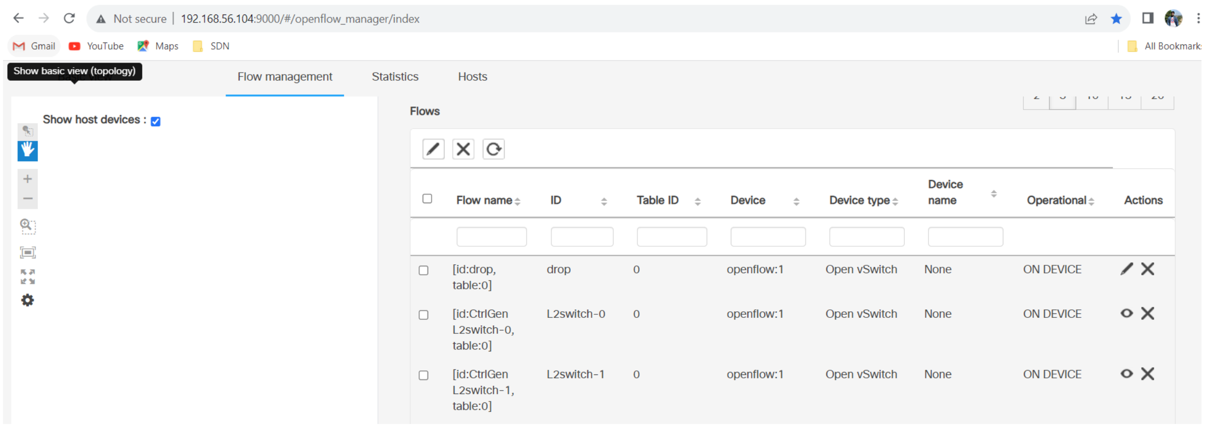


5. Paste the screenshot of your browser displaying Cisco OFM. It should display the Mininet topology. [**5 points]**



6. Create a flow entry on switch1 using Cisco OFM to provide the following functionality:
   a. Match the packets with source MAC address of host five and destination MAC address of host one. If there are any matches, then switch1 drops the corresponding packet.
   b. Set the priority of the flow entry as 50 and cookie value as 0x99.
   c. Paste the screenshot of the flow entry that's created in Cisco OFM. **[10 points]** *(Hint: Use the Flow Management tab inside Cisco OFM)*

Flow entry added to OFM

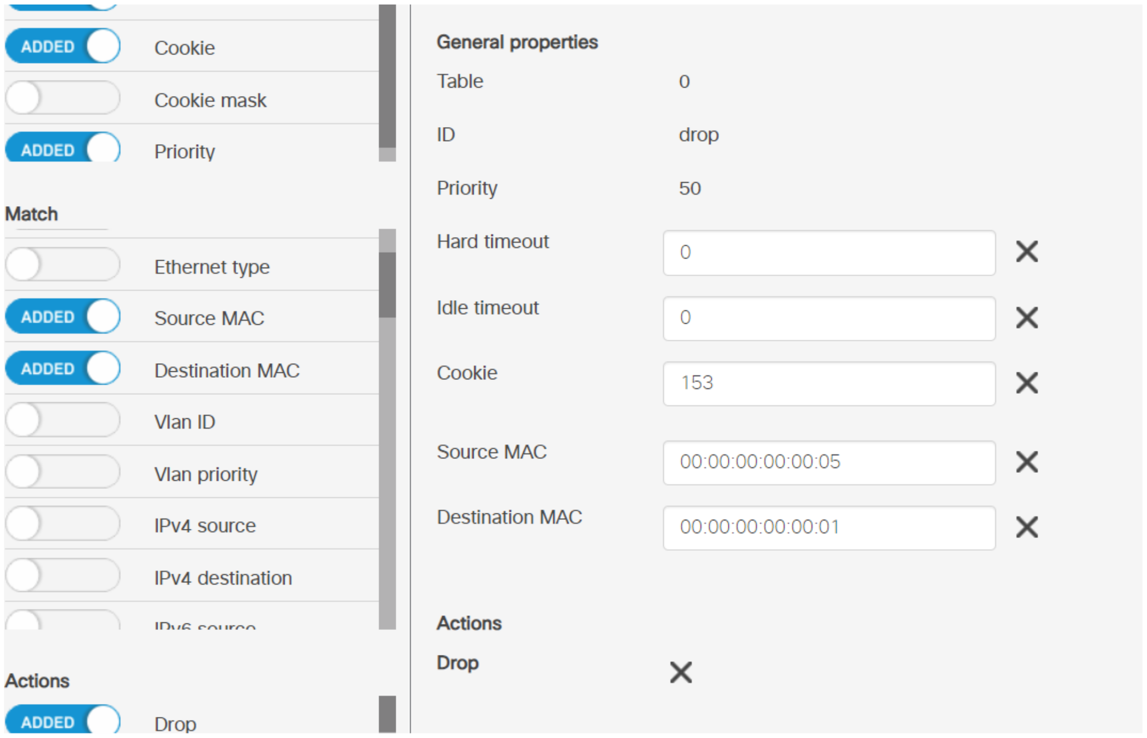


Not secure | 192.168.56.104:9000/#/openflow_manager/index
Gmail
YouTube
Maps
SDN
All Bookmarks
Show basic view (topology)
Flow management
Statistics
Hosts
Show host devices :

Flows

	Flow name	ID	Table ID	Device	Device type	Device name	Operational	Actions
☐	[id:drop, table:0]	drop	0	openflow:1	Open vSwitch	None	ON DEVICE	
☐	[id:CtrlGen L2switch-0, table:0]	L2switch-0	0	openflow:1	Open vSwitch	None	ON DEVICE	
☐	[id:CtrlGen L2switch-1, table:0]	L2switch-1	0	openflow:1	Open vSwitch	None	ON DEVICE	

Form to create flow entry



ADDED	Cookie
	Cookie mask
ADDED	Priority

Match

	Ethernet type
ADDED	Source MAC
ADDED	Destination MAC
	Vlan ID
	Vlan priority
	IPv4 source
	IPv4 destination
	IPv6 source

Actions

ADDED	Drop

General properties

Table	0
ID	drop
Priority	50
Hard timeout	0
Idle timeout	0
Cookie	153
Source MAC	00:00:00:00:00:05
Destination MAC	00:00:00:00:00:01

Actions

Drop

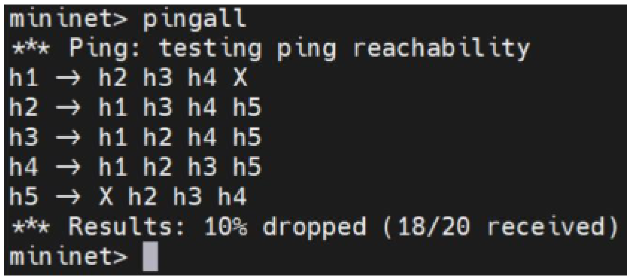
7. Explain the significance of priority value and cookie value of a flow entry. **[5 points]**

   Cookies - Cookies are used to map flows to applications that installed those flows, they play a significant role as it would help back track which flows were installed by which application/tenant. They provide metadata of the flows.

   Priority - Priority allows to match traffic based on its importance to a certain flow, a traffic from the same source should be treated differently if its a Video data stream or if its a regular browsing. These traffic for video applications should have higher priority allowing it to be processed in a certain way.
   Priority helps to specify, which match takes precedence, if a traffic could match multiple flows.

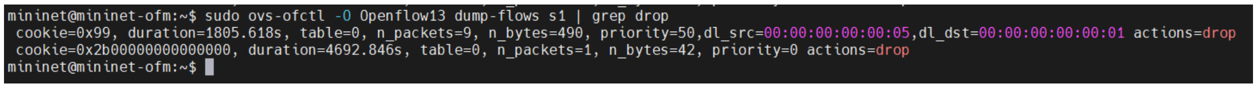
8. Issue pingall command in Mininet. Did all the hosts ping each other? If yes, why? If not, why? **[5 points]**

   Pingall works but, h1 is not able to ping h5, since OFM flow has been configured to drop packets.



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 X
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> X h2 h3 h4
*** Results: 10% dropped (18/20 received)
mininet>
```

9. From the Mininet VM, paste the screenshot of the command and corresponding output that shows the flow table highlighting the flow added by OFM. **[10 points]**



```
mininet@mininet-ofm:~$ sudo ovs-ofctl -O Openflow13 dump-flows s1 | grep drop
 cookie=0x99, duration=1805.618s, table=0, n_packets=9, n_bytes=490, priority=50,dl_src=00:00:00:00:00:05,dl_dst=00:00:00:00:00:01 actions=drop
 cookie=0x2b00000000000000, duration=4692.846s, table=0, n_packets=1, n_bytes=42, priority=0 actions=drop
mininet@mininet-ofm:~$
```

10. Explain the "flow logic" of how the Cisco OFM application is able to add OpenFlow flow table entries to the switches within Mininet. [**10 points**]

    ODL uses YANG models to represent network topologies and to specify how flows are present across the network. MD-SAL, which is a Model drive service adaption Layer consumes these models and abstracts them Northbound to the Apps as a REST API.
    OFM - OpenFlow Manager is an application that runs on the top of the ODL controller consuming these Northbound exposed REST APIs to collect flows from each switch.

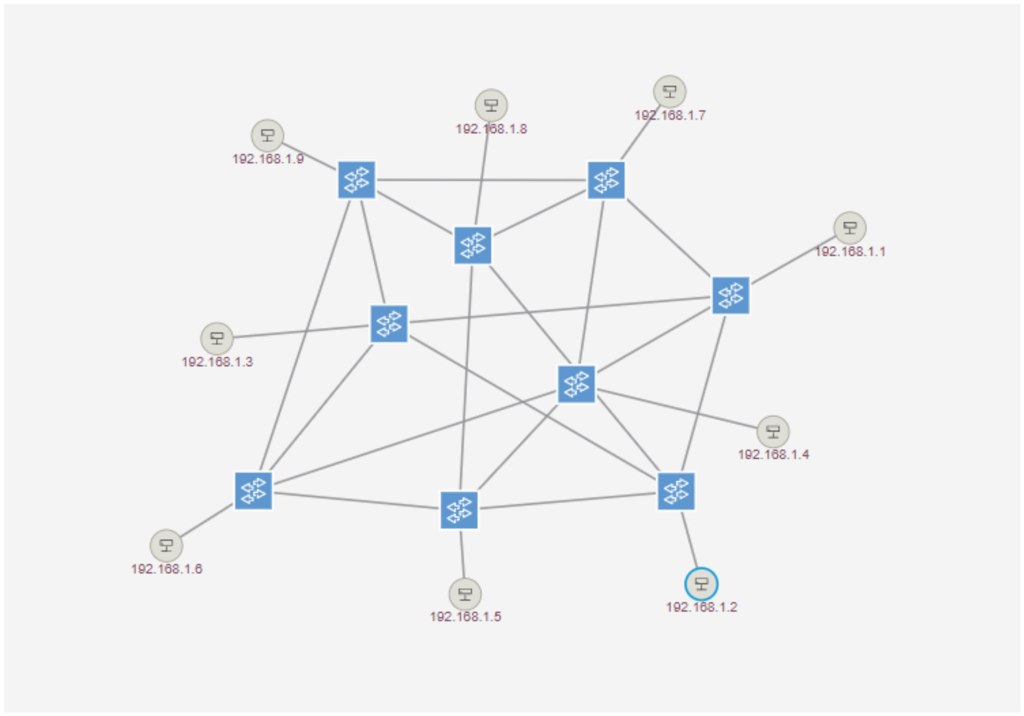
    Additionally, switches do run Openflow agents which are then used by the controller to perform operations such as adding a flow,etc.

Objective 2 - Connect Mininet to ONOS

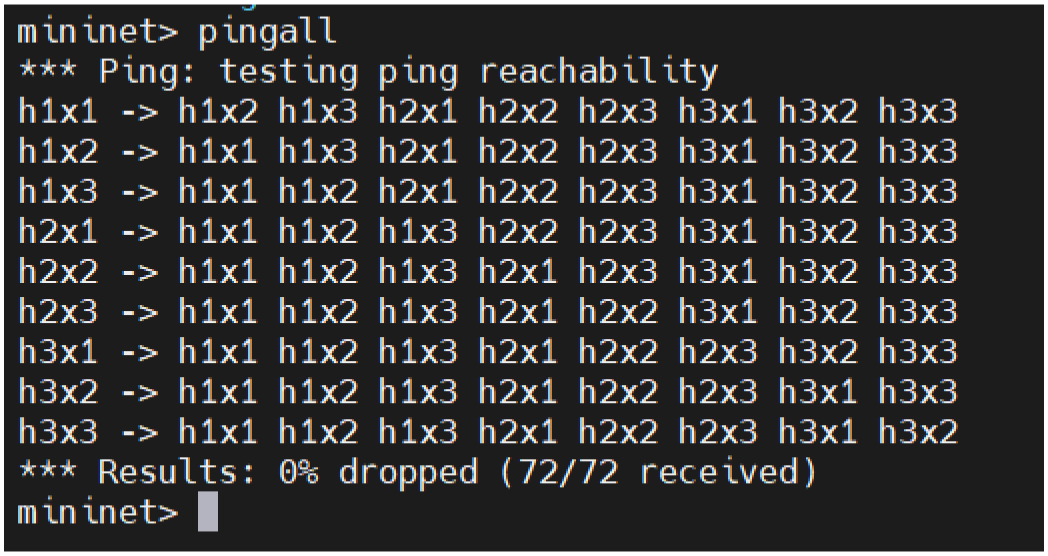
1. Refer the Lab 0 document to initialize ONOS.
2. Follow the instructions from the Lab 0 document to install apps and activate them.
3. Then start a Mininet topology that connects to a remote controller, uses easy to read MAC addresses, uses OpenFlow v1.3, and uses a network topology that is a torus with 3 and 3, and change the IP address scheme from the default to 192.168.1.0/24.
    a. Explain in detail what the torus topology is including the additional "3" settings. **[5 points]**

    Torus topology, or toroidal topology, is a network structure resembling a doughnut, offering high connectivity, fault tolerance, and efficient node interconnection. It is used to resemble dense networks.

    The settings could be
    - Basic Torus topology
      Nodes are ordered in 2D or 3D grid offering more efficient connectivity
    - 2D Torus Topology
      Node are connected in horizontal and vertical connections to allow parallel processing
    - 3D Torus Topology
      Nodes are connected horizontally, vertically, to back and to the front, used in supercomputers for low latency communication



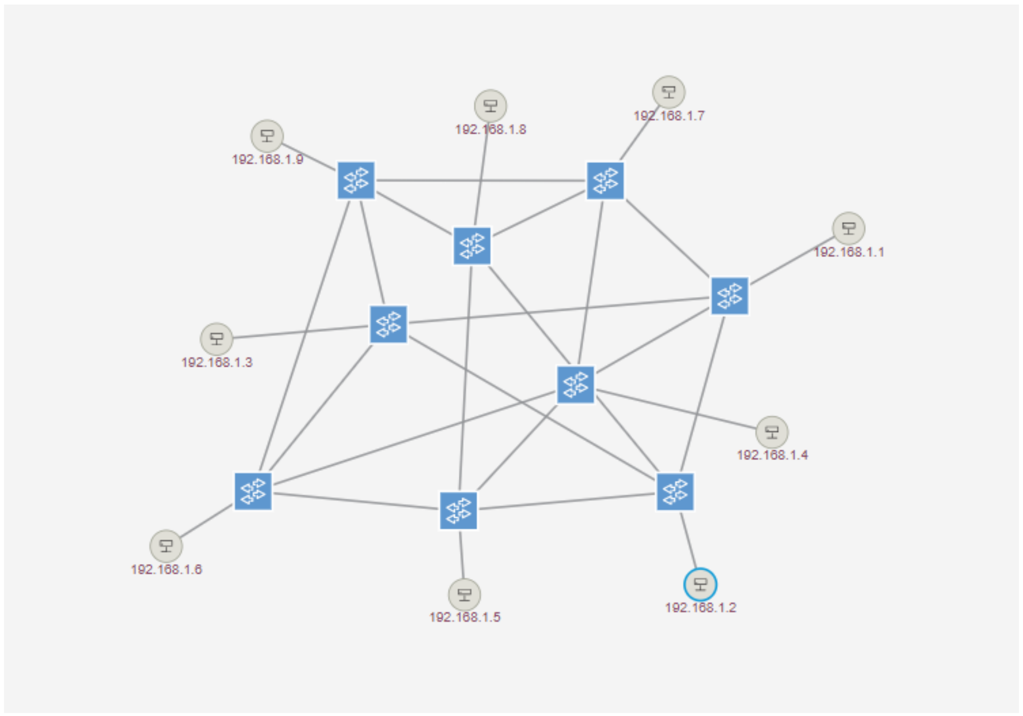
b. Issue the “pingall” command in Mininet to bring up all the devices.



```
mininet> pingall
*** Ping: testing ping reachability
h1x1 -> h1x2 h1x3 h2x1 h2x2 h2x3 h3x1 h3x2 h3x3
h1x2 -> h1x1 h1x3 h2x1 h2x2 h2x3 h3x1 h3x2 h3x3
h1x3 -> h1x1 h1x2 h2x1 h2x2 h2x3 h3x1 h3x2 h3x3
h2x1 -> h1x1 h1x2 h1x3 h2x2 h2x3 h3x1 h3x2 h3x3
h2x2 -> h1x1 h1x2 h1x3 h2x1 h2x3 h3x1 h3x2 h3x3
h2x3 -> h1x1 h1x2 h1x3 h2x1 h2x2 h3x1 h3x2 h3x3
h3x1 -> h1x1 h1x2 h1x3 h2x1 h2x2 h2x3 h3x2 h3x3
h3x2 -> h1x1 h1x2 h1x3 h2x1 h2x2 h2x3 h3x1 h3x3
h3x3 -> h1x1 h1x2 h1x3 h2x1 h2x2 h2x3 h3x1 h3x2
*** Results: 0% dropped (72/72 received)
mininet> 
```

4. Within the GUI of ONOS:

   a. Provide a screenshot of the topology (from the previous Mininet commands, including all hosts). **[10 points]**



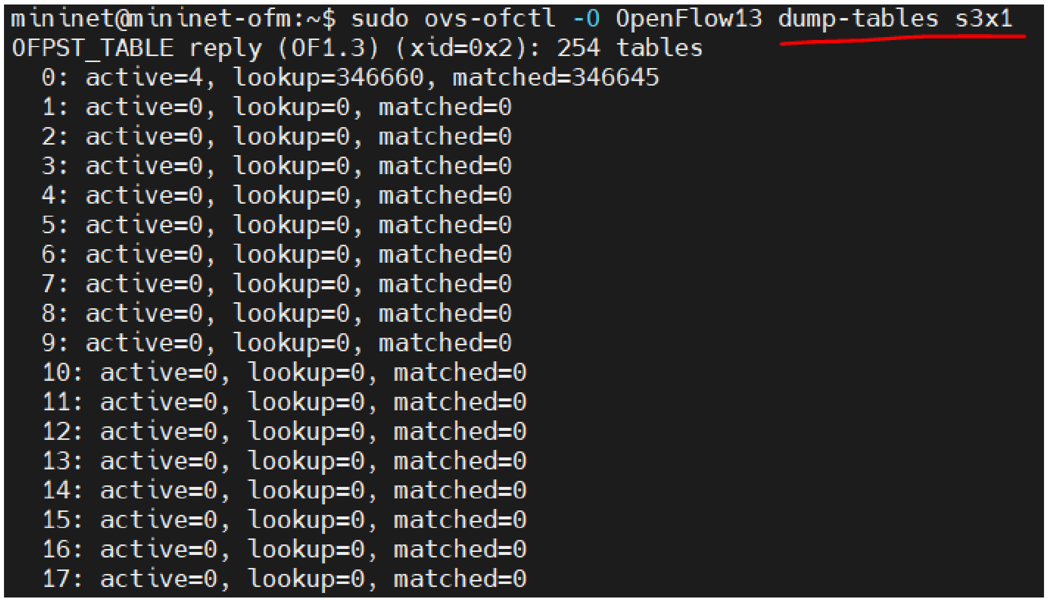
b. Navigate around the ONOS GUI and familiarize yourself with the features.

   i. Follow the "ONOS Tutorial" link on the desktop for additional information.

5. Within the Mininet VM, you are going to issue Open vSwitch commands to view statistics and flow table entries of the switches.

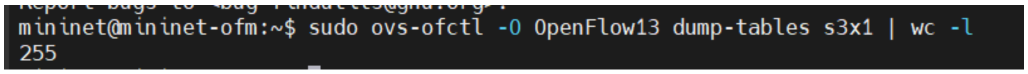
   a. Dump all OpenFlow flow tables from Switch3.

      i. What command did you use? How many flow tables were there? How do you know? What is the difference between flow tables and flow entries **[20 points]**



```
mininet@mininet-ofm:~$ sudo ovs-ofctl -O OpenFlow13 dump-tables s3x1
OFPST_TABLE reply (OF1.3) (xid=0x2): 254 tables
  0: active=4, lookup=346660, matched=346645
  1: active=0, lookup=0, matched=0
  2: active=0, lookup=0, matched=0
  3: active=0, lookup=0, matched=0
  4: active=0, lookup=0, matched=0
  5: active=0, lookup=0, matched=0
  6: active=0, lookup=0, matched=0
  7: active=0, lookup=0, matched=0
  8: active=0, lookup=0, matched=0
  9: active=0, lookup=0, matched=0
  10: active=0, lookup=0, matched=0
  11: active=0, lookup=0, matched=0
  12: active=0, lookup=0, matched=0
  13: active=0, lookup=0, matched=0
  14: active=0, lookup=0, matched=0
  15: active=0, lookup=0, matched=0
  16: active=0, lookup=0, matched=0
  17: active=0, lookup=0, matched=0
```

Excluding first line total tables = 255-1 =254



```
mininet@mininet-ofm:~$ sudo ovs-ofctl -O OpenFlow13 dump-tables s3x1 | wc -l
255
```

Difference between flow tables and flow entries

Flow table: **serves as a repository of rules and instructions for an SDN device, like a switch or router, on how to handle network packets.**

These tables are organized by specific functions such as forwarding, routing, or access control. Each flow table contains multiple flow entries, each of which defines a particular condition and action for packet processing.

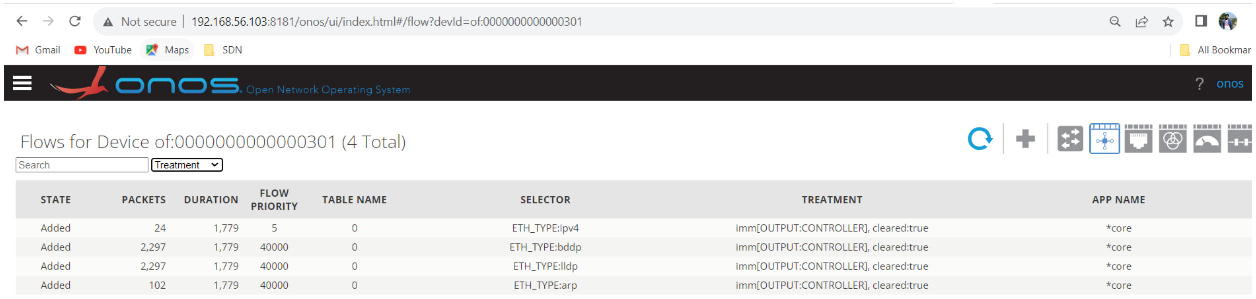
Flow entry: **It is a single rule within a flow table and outlines the criteria for matching packets, like source and destination IP addresses, port numbers, or protocol type, along with the actions to be taken when packets meet those conditions.**

These actions can include forwarding packets to a designated port, altering packet headers, discarding packets, or sending them to a controller for further analysis. Flow

entries within a flow table offer granularity, allowing network administrators to specify precisely how different types of traffic should be handled

Provide this same information by using the ONOS GUI.

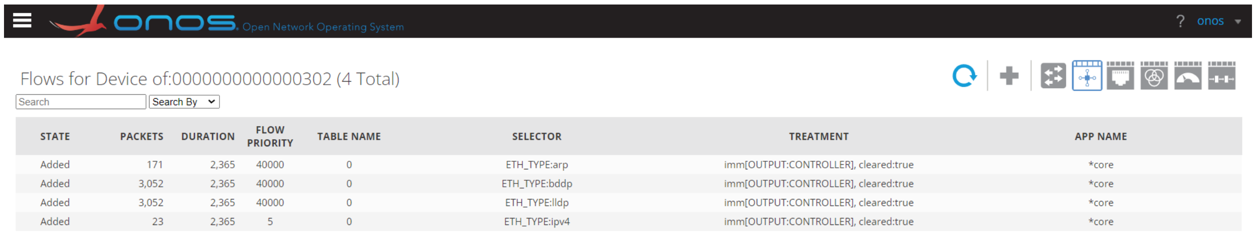
iii. Provide a screenshot of the flow tables and flow entries from Switch3 from the GUI. **[10 points]**



Not secure | 192.168.56.103:8181/onos/ui/index.html#/flow?devId=of:0000000000000301

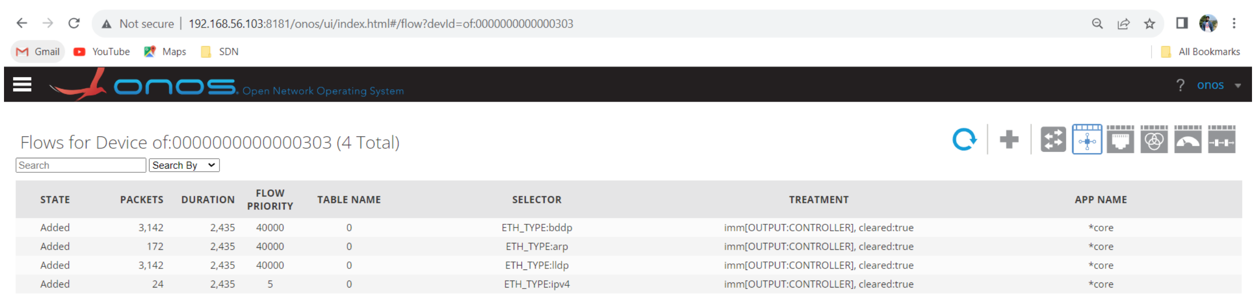
Flows for Device of:0000000000000301 (4 Total)

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	24	1,779	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	2,297	1,779	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	2,297	1,779	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	102	1,779	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core



Flows for Device of:0000000000000302 (4 Total)

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	171	2,365	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	3,052	2,365	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	3,052	2,365	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	23	2,365	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core



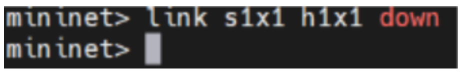
Not secure | 192.168.56.103:8181/onos/ui/index.html#/flow?devId=of:0000000000000303

Flows for Device of:0000000000000303 (4 Total)

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	3,142	2,435	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	172	2,435	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	3,142	2,435	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	24	2,435	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core

6. View the ONOS topology.
   a. Shutdown the link from Switch1 to Host1 (from within Mininet).
      i. What command did you use? **[5 points]**

         link s1x1 h1x1 down



```
mininet> link s1x1 h1x1 down
mininet> 
```

      ii. Did the ONOS GUI topology change in real-time or did you have to refresh? Provide two examples of how this Mininet command could be useful for testing. **[10 points]**

Yes, it will change in real time, and there is no need to refresh it.

This can be useful to stimulate link failures for an ongoing traffic, to check how the topology behaves to link failures, and responsiveness of ONOS to such changes. This can be effective in checking the resiliency of the network managed by the controller.

Another example would be to write a network healing App, and once the mininet link down command is issued, the App should be triggered and perform measures to report and take recovery actions and fix it. This can be tested via the command.

b. Bring the link back up from Switch1 to Host 1.

   i. What command did you use? **[2 points]**

   link s1x1 h1x1 up

7. Navigate around the ONOS GUI.

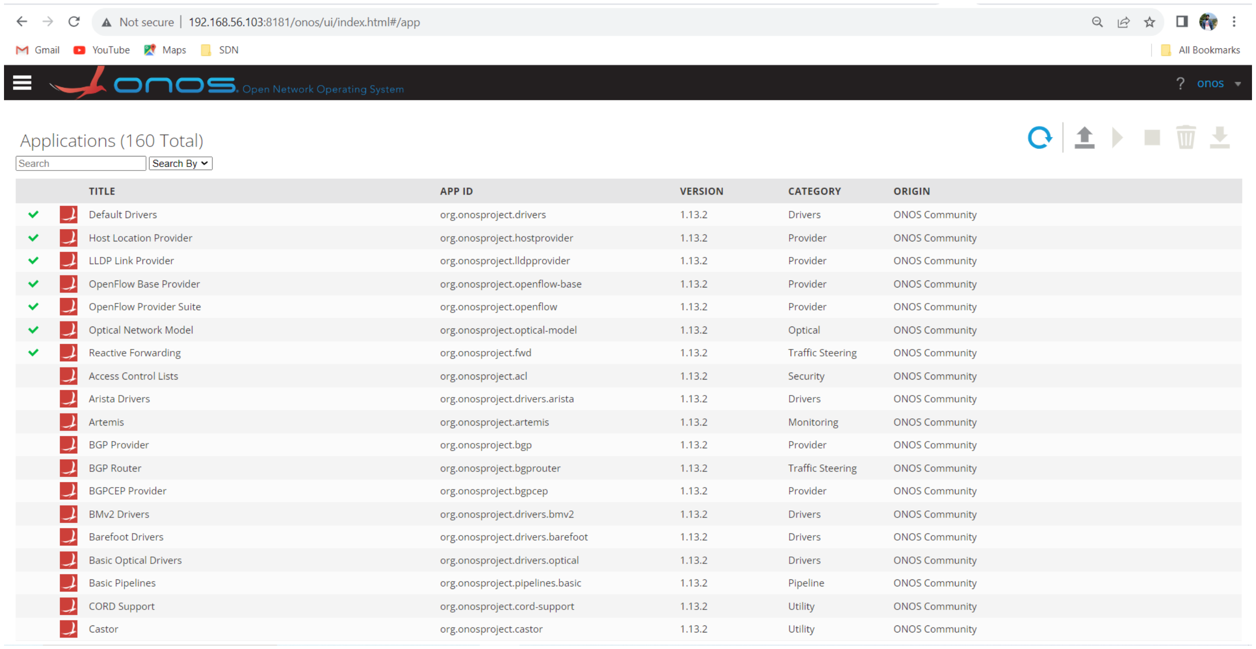
   a. Go to the main ONOS Platform menu.

      i. Then navigate to Applications.

      ii. Explain how you could activate additional applications and provide a screenshot as an example. **[10 points]**

1. Click on Menu on the top right corner
2. Go to Applications. Once we click on Applications, we see all the apps

   Apps with Green color are enabled.
3. To enable an Application, select the application and click on the play button/start on the top right corner.


Not secure | 192.168.56.103:8181/onos/ui/index.html#/app
Gmail YouTube Maps SDN All Bookmarks
Open Network Operating System
? onos
Search
Search By

Applications (160 Total)

	TITLE	APP ID	VERSION	CATEGORY	ORIGIN
✓	Default Drivers	org.onosproject.drivers	1.13.2	Drivers	ONOS Community
✓	Host Location Provider	org.onosproject.hostprovider	1.13.2	Provider	ONOS Community
✓	LLDP Link Provider	org.onosproject.lldpprovider	1.13.2	Provider	ONOS Community
✓	OpenFlow Base Provider	org.onosproject.openflow-base	1.13.2	Provider	ONOS Community
✓	OpenFlow Provider Suite	org.onosproject.openflow	1.13.2	Provider	ONOS Community
✓	Optical Network Model	org.onosproject.optical-model	1.13.2	Optical	ONOS Community
✓	Reactive Forwarding	org.onosproject.fwd	1.13.2	Traffic Steering	ONOS Community
	Access Control Lists	org.onosproject.acl	1.13.2	Security	ONOS Community
	Arista Drivers	org.onosproject.drivers.arista	1.13.2	Drivers	ONOS Community
	Artemis	org.onosproject.artemis	1.13.2	Monitoring	ONOS Community
	BGP Provider	org.onosproject.bgp	1.13.2	Provider	ONOS Community
	BGP Router	org.onosproject.bgprouter	1.13.2	Traffic Steering	ONOS Community
	BGPCEP Provider	org.onosproject.bgpcep	1.13.2	Provider	ONOS Community
	BMv2 Drivers	org.onosproject.drivers.bmv2	1.13.2	Drivers	ONOS Community
	Barefoot Drivers	org.onosproject.drivers.barefoot	1.13.2	Drivers	ONOS Community
	Basic Optical Drivers	org.onosproject.drivers.optical	1.13.2	Drivers	ONOS Community
	Basic Pipelines	org.onosproject.pipelines.basic	1.13.2	Pipeline	ONOS Community
	CORD Support	org.onosproject.cord-support	1.13.2	Utility	ONOS Community
	Castor	org.onosproject.castor	1.13.2	Utility	ONOS Community

b. Intentions:

i. What are intentions in ONOS? **[5 points]**

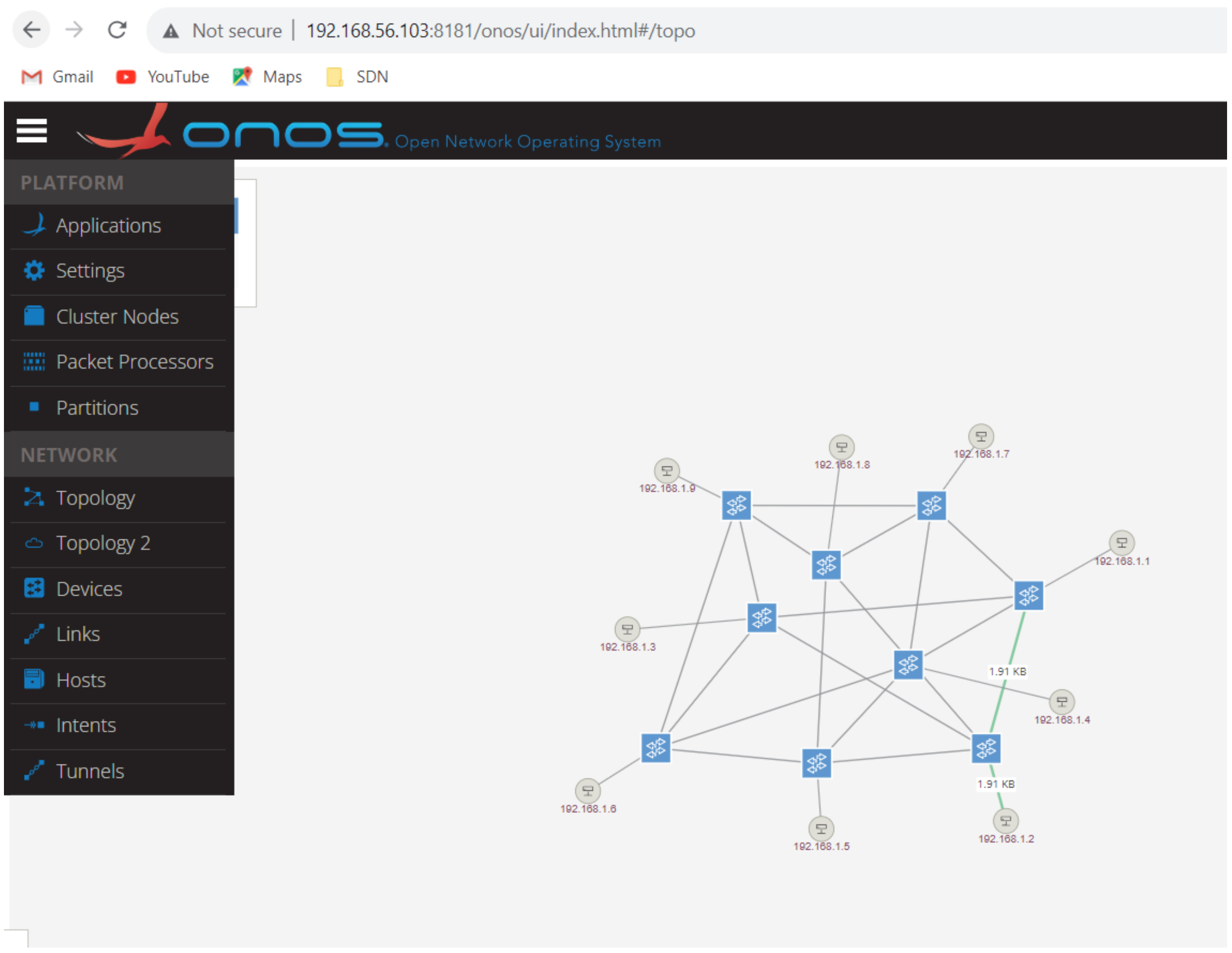
Intentions refer to high-level declarative policies that define the desired network behavior and configurations.

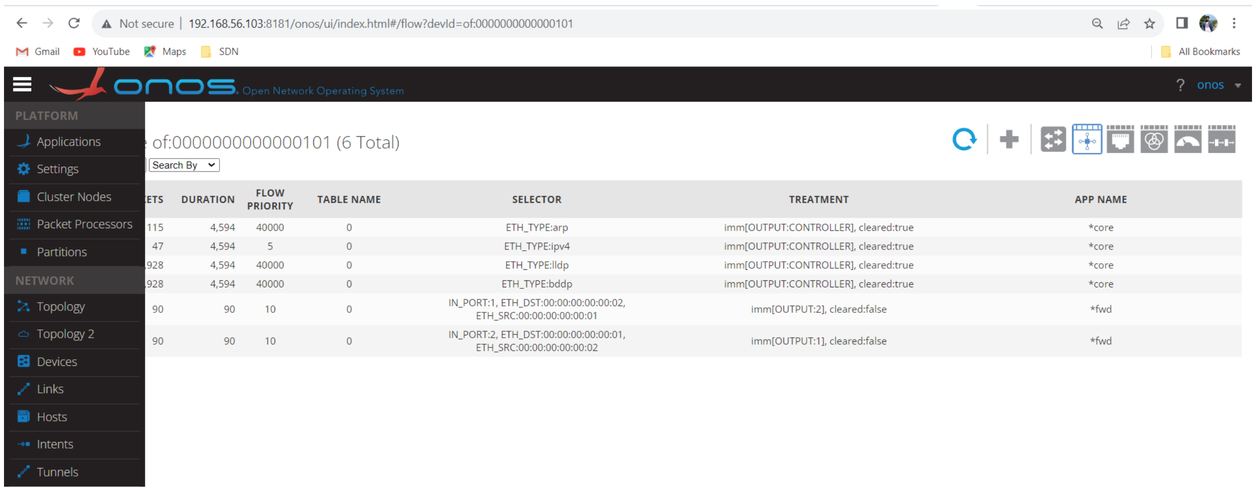
It is used to abstract and simplify network management tasks instead of manually configuring individual network devices, admins can express their intentions, such as connectivity requirements or security policies, in a more human-readable and abstract manner.

ONOS then translates these intentions into specific network configurations and enforces them across the network infrastructure.

ii. Create a Host to Host flow between two hosts in the topology.

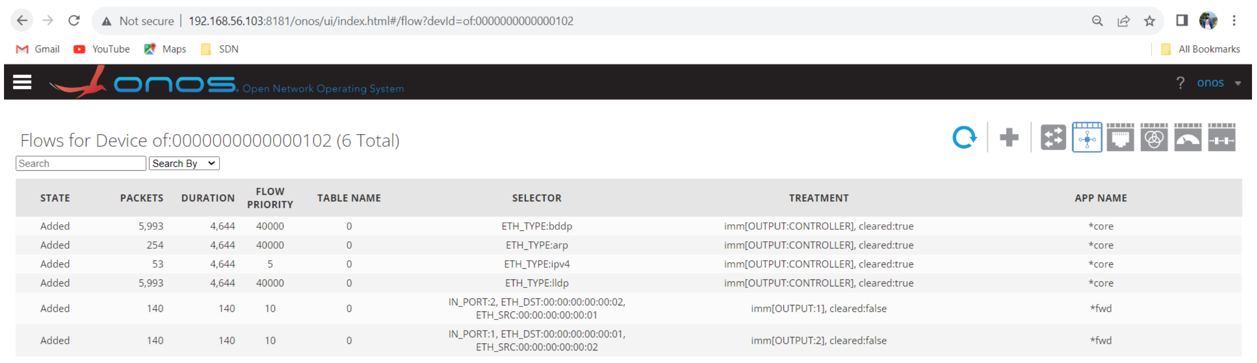
1. Generate traffic between these two hosts.
2. Provide a screenshot of the flow as well as the green link indicating the bandwidth on that flow in the GUI. **[15 points]**





of:0000000000000101 (6 Total)

ETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
115	4,594	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
47	4,594	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
928	4,594	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
928	4,594	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
90	90	10	0	IN_PORT:1, ETH_DST:00:00:00:00:00:02, ETH_SRC:00:00:00:00:00:01	imm[OUTPUT:2], cleared:false	*fwd
90	90	10	0	IN_PORT:2, ETH_DST:00:00:00:00:00:01, ETH_SRC:00:00:00:00:00:02	imm[OUTPUT:1], cleared:false	*fwd



Not secure | 192.168.56.103:8181/onos/ui/index.html#/flow?devId=of:0000000000000102

Flows for Device of:0000000000000102 (6 Total)

STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Added	5,993	4,644	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	254	4,644	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	53	4,644	5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	5,993	4,644	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Added	140	140	10	0	IN_PORT:2, ETH_DST:00:00:00:00:00:02, ETH_SRC:00:00:00:00:00:01	imm[OUTPUT:1], cleared:false	*fwd
Added	140	140	10	0	IN_PORT:1, ETH_DST:00:00:00:00:00:01, ETH_SRC:00:00:00:00:00:02	imm[OUTPUT:2], cleared:false	*fwd

3. If you are having trouble with this based on this topology, you can ping from a host to another host in Mininet and then "show related traffic" in the GUI after clicking on the host you initiated the ping from. Then you can use the "a" shortcut to show all traffic flows with bandwidth.
4. Based on the knowledge you have thus far, provide three functionalities/readability you can do in the ONOS controller that you can't do in ODL. **[10 points]**

Intent Based Networking -

In ODL, we cannot simply specify the intent and actual specification needs to be provided per switch to install flows. Intent based networking can be useful to abstract the entire network.

More real-time -

A link failure in ODL required to click reload to realize the network changes. However, in ONOS these changes are in realtime.

Efficiency and Scalability -

Creating a mininet linear topology with 100 switches takes forever for ODL to come up and ONOS starts showing the devices the moment few devices are in sync with the controller.

5. With your experience thus far, do you like the ONOS or ODL controller better? Why? **[5 points]**

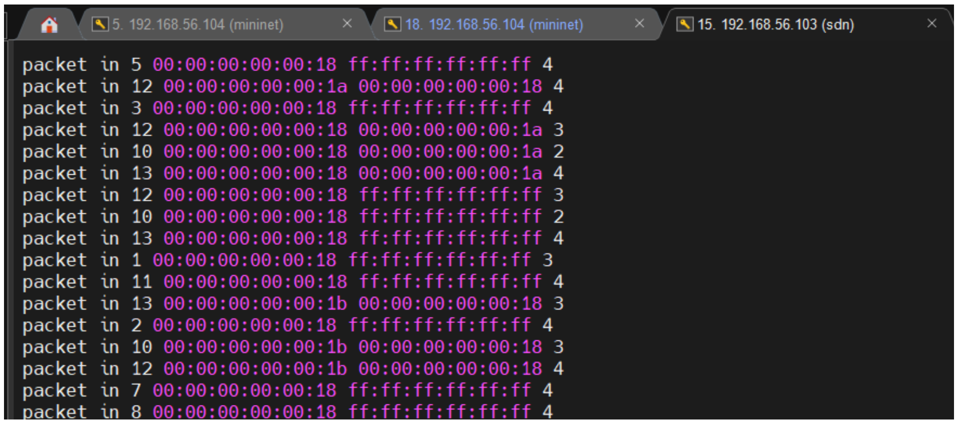
ONOS is far better than ODL, since I didn't have to enable a number of features for regular switching. ONOS might be less buggy, since ODL most of the features are removed and are managed separately.

For huge networks, ONOS works better and gives real time GUI reactiveness which makes the controller useful to detect issues in the network and react quickly.

Intent based configuration provides more abstraction to network administrators, which can help manage complex networks with ease.

Objective 3 - Connect Mininet to Ryu

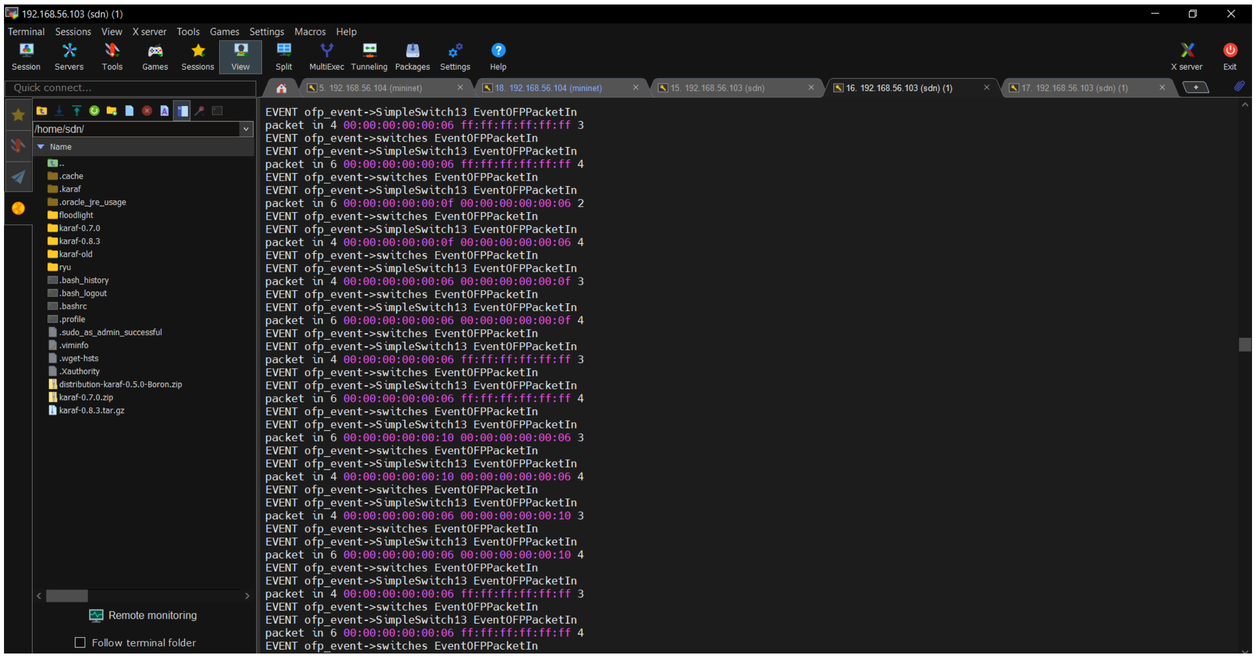
1. Refer the Lab 0 document to initialize Ryu.
2. Enable the Ryu web interface with a basic L2 switch (use the Lab 0 document for assistance).
3. Login to the Ryu web interface.
4. Login to Mininet.
    a. Start a new Mininet topology that connects to a remote controller (Ryu), uses easy to read MAC addresses, and uses a network topology that is a tree with a fanout of three and depth of three.
    b. Issue the "pingall" command in Mininet to bring up all the devices.
        i. What messages do you see on the Ryu CLI debug? Indicate why you see these messages. **[5 points]**



```
packet in 5 00:00:00:00:00:18 ff:ff:ff:ff:ff:ff 4
packet in 12 00:00:00:00:00:1a 00:00:00:00:00:18 4
packet in 3 00:00:00:00:00:18 ff:ff:ff:ff:ff:ff 4
packet in 12 00:00:00:00:00:18 00:00:00:00:00:1a 3
packet in 10 00:00:00:00:00:18 00:00:00:00:00:1a 2
packet in 13 00:00:00:00:00:18 00:00:00:00:00:1a 4
packet in 12 00:00:00:00:00:18 ff:ff:ff:ff:ff:ff 3
packet in 10 00:00:00:00:00:18 ff:ff:ff:ff:ff:ff 2
packet in 13 00:00:00:00:00:18 ff:ff:ff:ff:ff:ff 4
packet in 1 00:00:00:00:00:18 ff:ff:ff:ff:ff:ff 3
packet in 11 00:00:00:00:00:18 ff:ff:ff:ff:ff:ff 4
packet in 13 00:00:00:00:00:1b 00:00:00:00:00:18 3
packet in 2 00:00:00:00:00:18 ff:ff:ff:ff:ff:ff 4
packet in 10 00:00:00:00:00:1b 00:00:00:00:00:18 3
packet in 12 00:00:00:00:00:1b 00:00:00:00:00:18 4
packet in 7 00:00:00:00:00:18 ff:ff:ff:ff:ff:ff 4
packet in 8 00:00:00:00:00:18 ff:ff:ff:ff:ff:ff 4
```

Ryu controller prints the debug logs showing information about packet In messages.

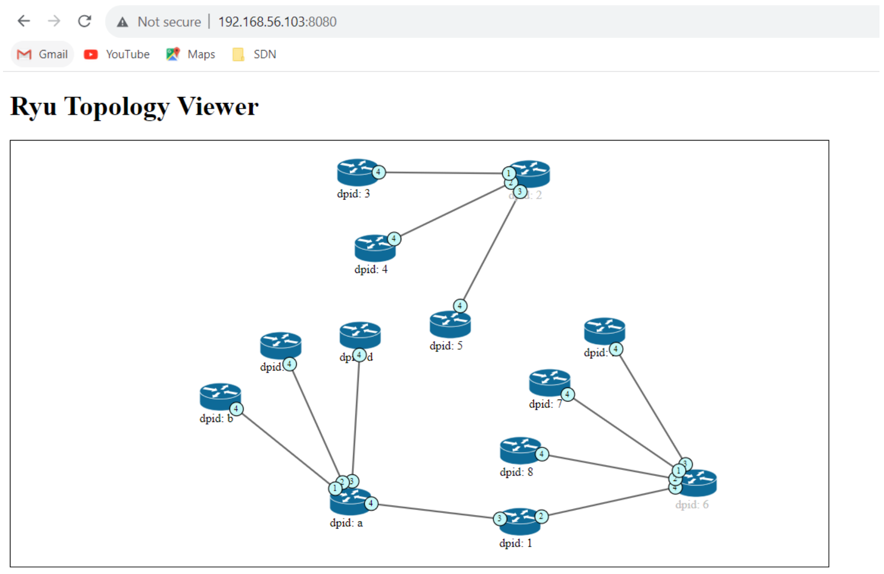
ofp_event represents a packet in message that is being processed



```
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 4 00:00:00:00:00:06 ff:ff:ff:ff:ff:ff 3
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 6 00:00:00:00:00:06 ff:ff:ff:ff:ff:ff 4
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 6 00:00:00:00:00:0f 00:00:00:00:00:06 2
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 4 00:00:00:00:00:0f 00:00:00:00:00:06 4
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 4 00:00:00:00:00:06 00:00:00:00:00:0f 3
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 6 00:00:00:00:00:06 00:00:00:00:00:0f 4
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 4 00:00:00:00:00:06 ff:ff:ff:ff:ff:ff 3
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 6 00:00:00:00:00:06 ff:ff:ff:ff:ff:ff 4
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 6 00:00:00:00:00:10 00:00:00:00:00:06 3
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 4 00:00:00:00:00:10 00:00:00:00:00:06 4
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 4 00:00:00:00:00:06 00:00:00:00:00:10 3
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 6 00:00:00:00:00:06 00:00:00:00:00:10 4
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 4 00:00:00:00:00:06 ff:ff:ff:ff:ff:ff 3
EVENT ofp_event->switches EventOFPPacketIn
EVENT ofp_event->SimpleSwitch13 EventOFPPacketIn
packet in 6 00:00:00:00:00:06 ff:ff:ff:ff:ff:ff 4
EVENT ofp_event->switches EventOFPPacketIn
```

These are the debug level logs that are printed by the controller to show the operations that are performed.

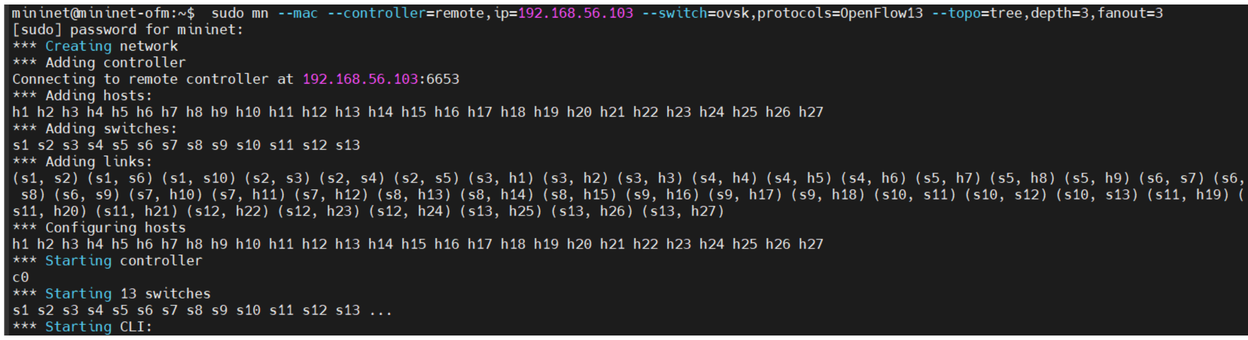
c. Within the GUI of Ryu, provide a screenshot of the topology. **[10 points]**



Objective 4 - Connect Mininet to Floodlight

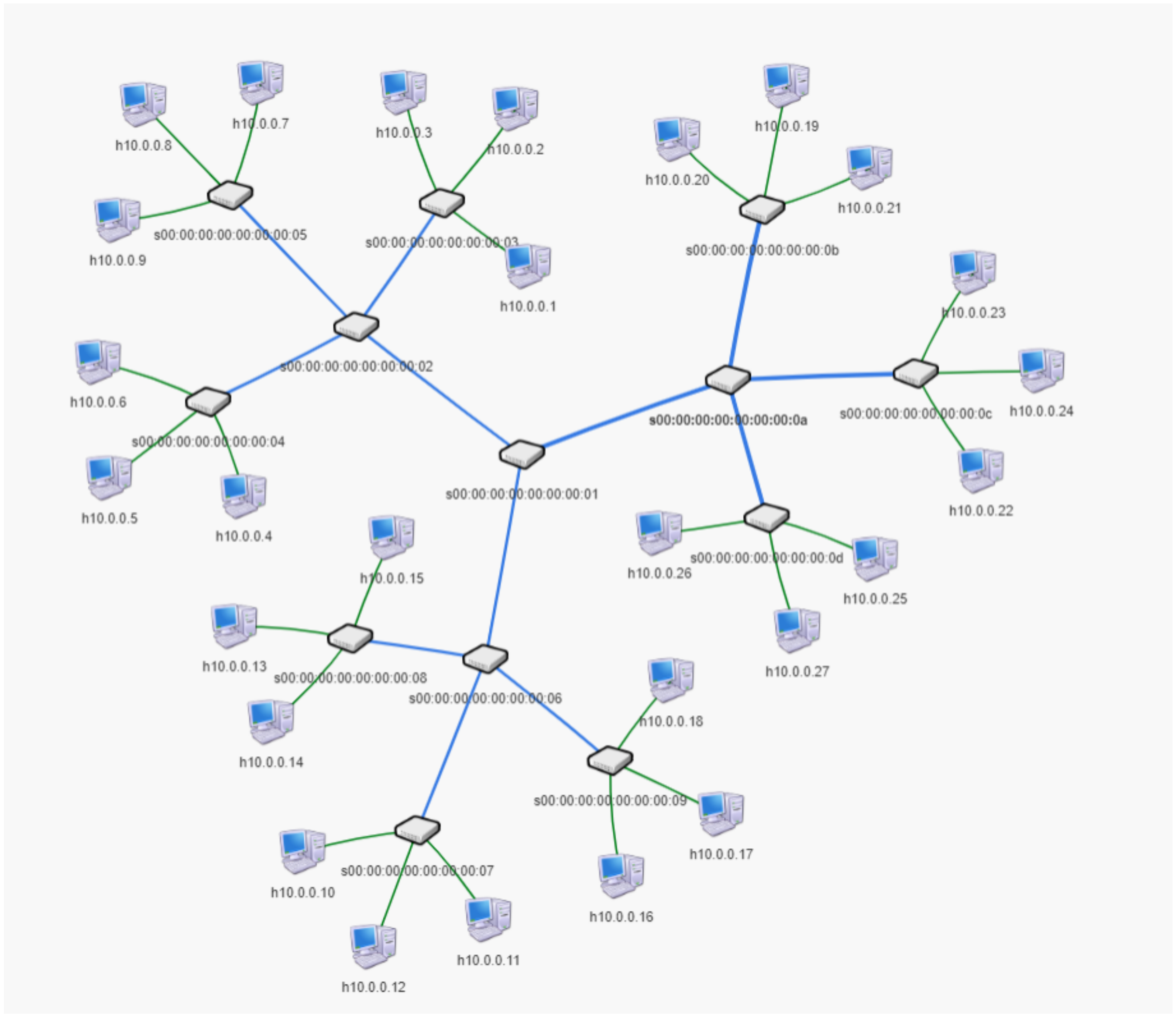
1. Refer to the Lab 0 document to initialize Floodlight.
2. Access the Floodlight Web UI.
3. Open a new terminal and get into the Floodlight directory. Now start Mininet, and build a tree topology with a depth of 3, a fanout of 3, using OpenFlow version 1.3, and port number 6653:
    a. What command did you use? **[5 points]**

sudo mn --mac --controller=remote,ip=192.168.56.103
--switch=ovsk,protocols=OpenFlow13 --topo=tree,depth=3,fanout=3

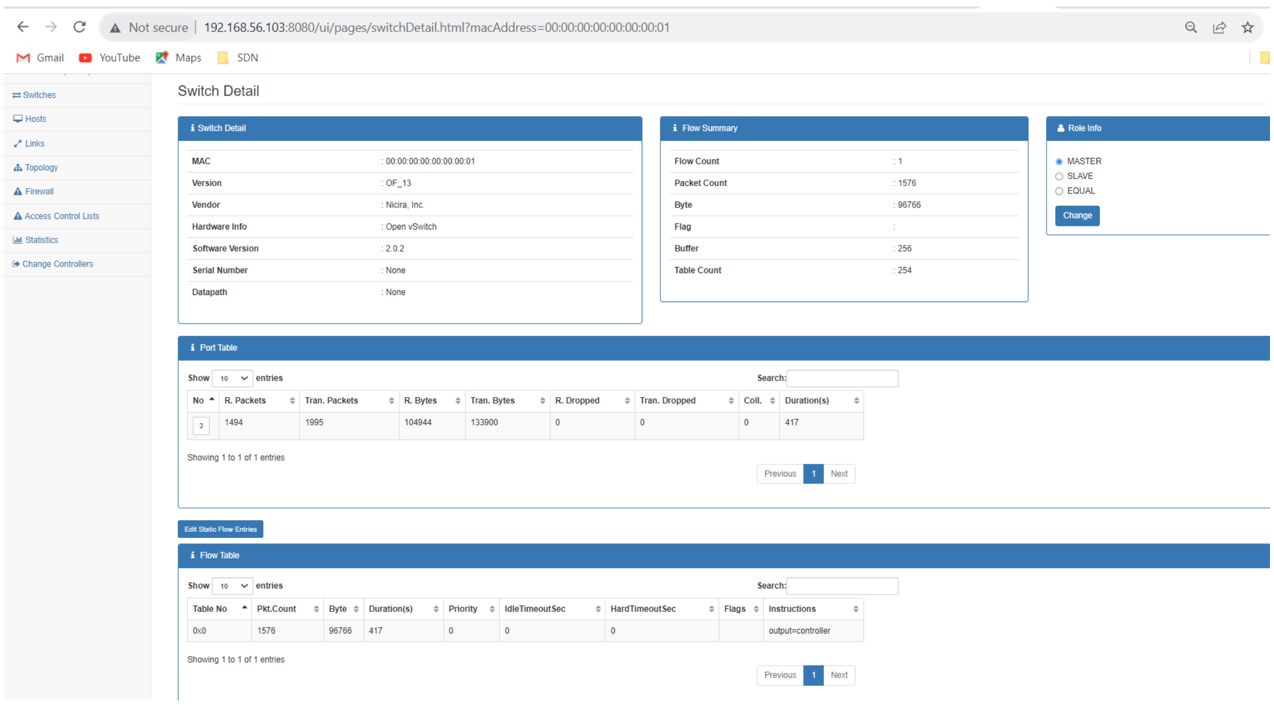


```
mininet@mininet-ofm:~$  sudo mn --mac --controller=remote,ip=192.168.56.103 --switch=ovsk,protocols=OpenFlow13 --topo=tree,depth=3,fanout=3
[sudo] password for mininet:
*** Creating network
*** Adding controller
Connecting to remote controller at 192.168.56.103:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13
*** Adding links:
(s1, s2) (s1, s6) (s1, s10) (s2, s3) (s2, s4) (s2, s5) (s3, h1) (s3, h2) (s3, h3) (s4, h4) (s4, h5) (s4, h6) (s5, h7) (s5, h8) (s5, h9) (s6, s7) (s6,
 s8) (s6, s9) (s7, h10) (s7, h11) (s7, h12) (s8, h13) (s8, h14) (s8, h15) (s9, h16) (s9, h17) (s9, h18) (s10, s11) (s10, s12) (s10, s13) (s11, h19) (
s11, h20) (s11, h21) (s12, h22) (s12, h23) (s12, h24) (s13, h25) (s13, h26) (s13, h27)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
*** Starting controller
c0
*** Starting 13 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 ...
*** Starting CLI:
```

    b. Paste the screenshot of the topology created on the Floodlight controller dashboard. **[5 points]**



4. After the topology is created on your Floodlight Web UI, proceed to evaluate the 'Switch Details' on any switch you desire. Post a screenshot of the switch details screen. **[10 points]**



Not secure | 192.168.56.103:8080/ui/pages/switchDetail.html?macAddress=00:00:00:00:00:00:00:01

Gmail YouTube Maps SDN

Switches
Hosts
Links
Topology
Firewall
Access Control Lists
Statistics
Change Controllers

Switch Detail

Switch Detail

MAC	: 00:00:00:00:00:00:00:01
Version	: OF_13
Vendor	: Nicira, Inc.
Hardware Info	: Open vSwitch
Software Version	: 2.0.2
Serial Number	: None
Datapath	: None

Flow Summary

Flow Count	: 1
Packet Count	: 1576
Byte	: 96766
Flag	:
Buffer	: 256
Table Count	: 254

Role Info

[✓ MASTER] SLAVE EQUAL

Change

Port Table

Show 10 entries Search:

No	R. Packets	Tran. Packets	R. Bytes	Tran. Bytes	R. Dropped	Tran. Dropped	Coll.	Duration(s)
3	1494	1995	104944	133900	0	0	0	417

Showing 1 to 1 of 1 entries

Previous 1 Next

Edit Static Flow Entries

Flow Table

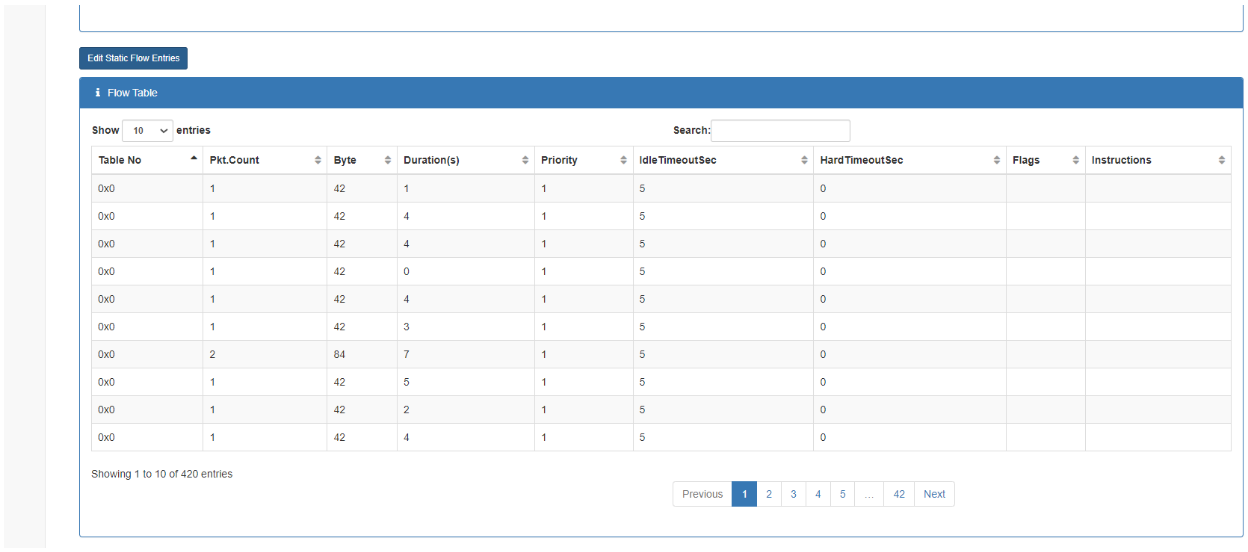
Show 10 entries Search:

Table No	Pkt.Count	Byte	Duration(s)	Priority	IdleTimeoutSec	HardTimeoutSec	Flags	Instructions
0x0	1576	96766	417	0	0	0		output=controller

Showing 1 to 1 of 1 entries

Previous 1 Next

After ping -



Edit Static Flow Entries

Flow Table

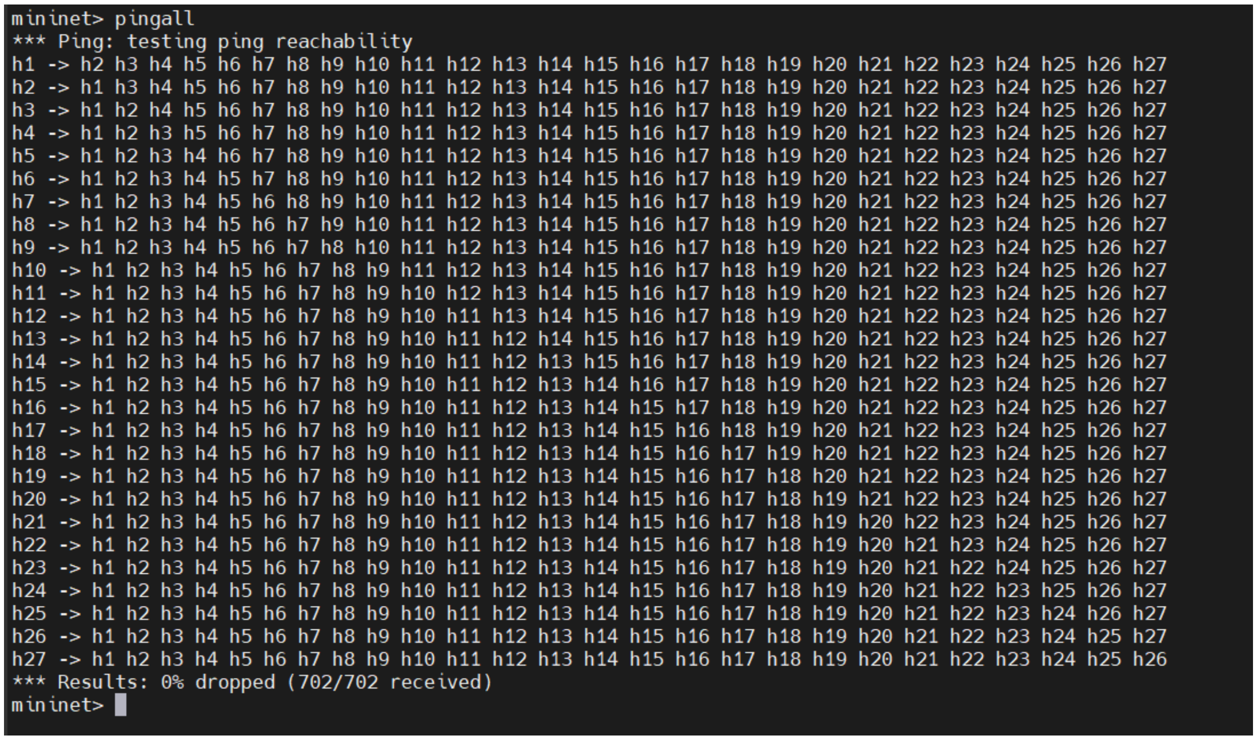
Show 10 entries Search:

Table No	Pkt.Count	Byte	Duration(s)	Priority	IdleTimeoutSec	HardTimeoutSec	Flags	Instructions
0x0	1	42	1	1	5	0		
0x0	1	42	4	1	5	0		
0x0	1	42	4	1	5	0		
0x0	1	42	0	1	5	0		
0x0	1	42	4	1	5	0		
0x0	1	42	3	1	5	0		
0x0	2	84	7	1	5	0		
0x0	1	42	5	1	5	0		
0x0	1	42	2	1	5	0		
0x0	1	42	4	1	5	0		

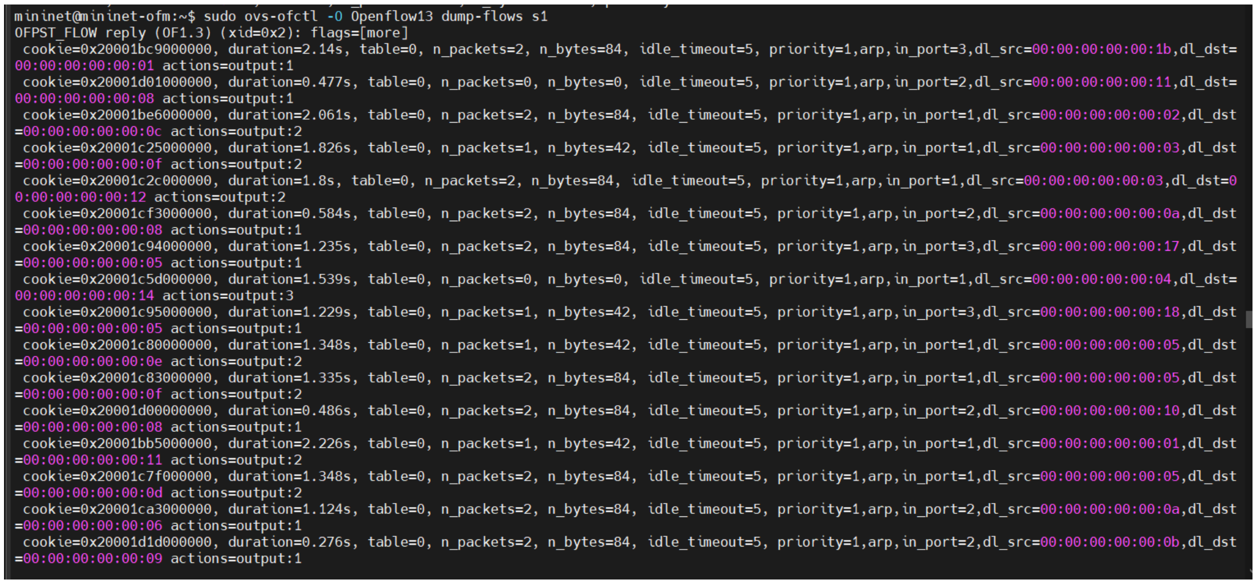
Showing 1 to 10 of 420 entries

Previous 1 2 3 4 5 … 42 Next

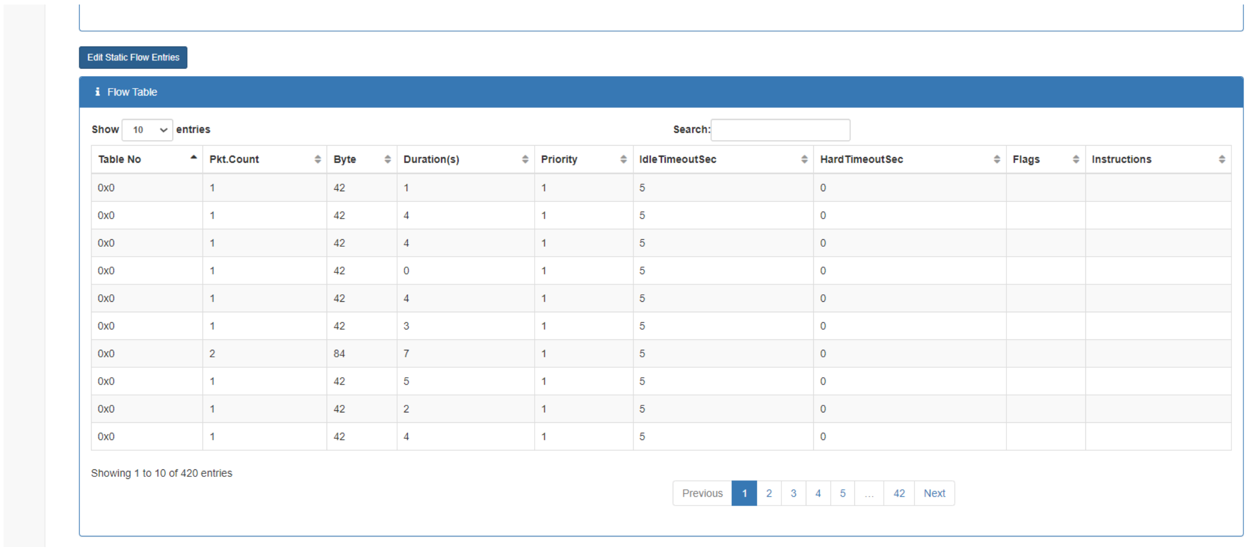
5. Confirm you have issued a 'pingall' operation on Mininet for the topology created and evaluate the flow table entries. Post a screenshot of the updated flow table entries. **[10 points]**



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h13 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h14 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h15 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h16 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h17 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h18 h19 h20 h21 h22 h23 h24 h25 h26 h27
h18 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h19 h20 h21 h22 h23 h24 h25 h26 h27
h19 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h20 h21 h22 h23 h24 h25 h26 h27
h20 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h21 h22 h23 h24 h25 h26 h27
h21 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h22 h23 h24 h25 h26 h27
h22 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h23 h24 h25 h26 h27
h23 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h24 h25 h26 h27
h24 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h25 h26 h27
h25 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h26 h27
h26 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h27
h27 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22 h23 h24 h25 h26
*** Results: 0% dropped (702/702 received)
mininet> 
```



```
mininet@mininet-ofm:~$ sudo ovs-ofctl -O Openflow13 dump-flows s1
OFPST_FLOW reply (OF1.3) (xid=0x2): flags=[more]
 cookie=0x20001bc9000000, duration=2.14s, table=0, n_packets=2, n_bytes=84, idle_timeout=5, priority=1,arp,in_port=3,dl_src=00:00:00:00:00:1b,dl_dst=
00:00:00:00:00:01 actions=output:1
 cookie=0x20001d01000000, duration=0.477s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,arp,in_port=2,dl_src=00:00:00:00:00:11,dl_dst=
00:00:00:00:00:08 actions=output:1
 cookie=0x20001be6000000, duration=2.061s, table=0, n_packets=2, n_bytes=84, idle_timeout=5, priority=1,arp,in_port=1,dl_src=00:00:00:00:00:02,dl_dst
=00:00:00:00:00:0c actions=output:2
 cookie=0x20001c25000000, duration=1.826s, table=0, n_packets=1, n_bytes=42, idle_timeout=5, priority=1,arp,in_port=1,dl_src=00:00:00:00:00:03,dl_dst
=00:00:00:00:00:0f actions=output:2
 cookie=0x20001c2c000000, duration=1.8s, table=0, n_packets=2, n_bytes=84, idle_timeout=5, priority=1,arp,in_port=1,dl_src=00:00:00:00:00:03,dl_dst=0
0:00:00:00:00:12 actions=output:2
 cookie=0x20001cf3000000, duration=0.584s, table=0, n_packets=2, n_bytes=84, idle_timeout=5, priority=1,arp,in_port=2,dl_src=00:00:00:00:00:0a,dl_dst
=00:00:00:00:00:08 actions=output:1
 cookie=0x20001c94000000, duration=1.235s, table=0, n_packets=2, n_bytes=84, idle_timeout=5, priority=1,arp,in_port=3,dl_src=00:00:00:00:00:17,dl_dst
=00:00:00:00:00:05 actions=output:1
 cookie=0x20001c5d000000, duration=1.539s, table=0, n_packets=0, n_bytes=0, idle_timeout=5, priority=1,arp,in_port=1,dl_src=00:00:00:00:00:04,dl_dst=
00:00:00:00:00:14 actions=output:3
 cookie=0x20001c95000000, duration=1.229s, table=0, n_packets=1, n_bytes=42, idle_timeout=5, priority=1,arp,in_port=3,dl_src=00:00:00:00:00:18,dl_dst
=00:00:00:00:00:05 actions=output:1
 cookie=0x20001c80000000, duration=1.348s, table=0, n_packets=1, n_bytes=42, idle_timeout=5, priority=1,arp,in_port=1,dl_src=00:00:00:00:00:05,dl_dst
=00:00:00:00:00:0e actions=output:2
 cookie=0x20001c83000000, duration=1.335s, table=0, n_packets=2, n_bytes=84, idle_timeout=5, priority=1,arp,in_port=1,dl_src=00:00:00:00:00:05,dl_dst
=00:00:00:00:00:0f actions=output:2
 cookie=0x20001d00000000, duration=0.486s, table=0, n_packets=2, n_bytes=84, idle_timeout=5, priority=1,arp,in_port=2,dl_src=00:00:00:00:00:10,dl_dst
=00:00:00:00:00:08 actions=output:1
 cookie=0x20001bb5000000, duration=2.226s, table=0, n_packets=1, n_bytes=42, idle_timeout=5, priority=1,arp,in_port=1,dl_src=00:00:00:00:00:01,dl_dst
=00:00:00:00:00:11 actions=output:2
 cookie=0x20001c7f000000, duration=1.348s, table=0, n_packets=2, n_bytes=84, idle_timeout=5, priority=1,arp,in_port=1,dl_src=00:00:00:00:00:05,dl_dst
=00:00:00:00:00:0d actions=output:2
 cookie=0x20001ca3000000, duration=1.124s, table=0, n_packets=2, n_bytes=84, idle_timeout=5, priority=1,arp,in_port=2,dl_src=00:00:00:00:00:0a,dl_dst
=00:00:00:00:00:06 actions=output:1
 cookie=0x20001d1d000000, duration=0.276s, table=0, n_packets=2, n_bytes=84, idle_timeout=5, priority=1,arp,in_port=2,dl_src=00:00:00:00:00:0b,dl_dst
=00:00:00:00:00:09 actions=output:1
```



Edit Static Flow Entries

Flow Table

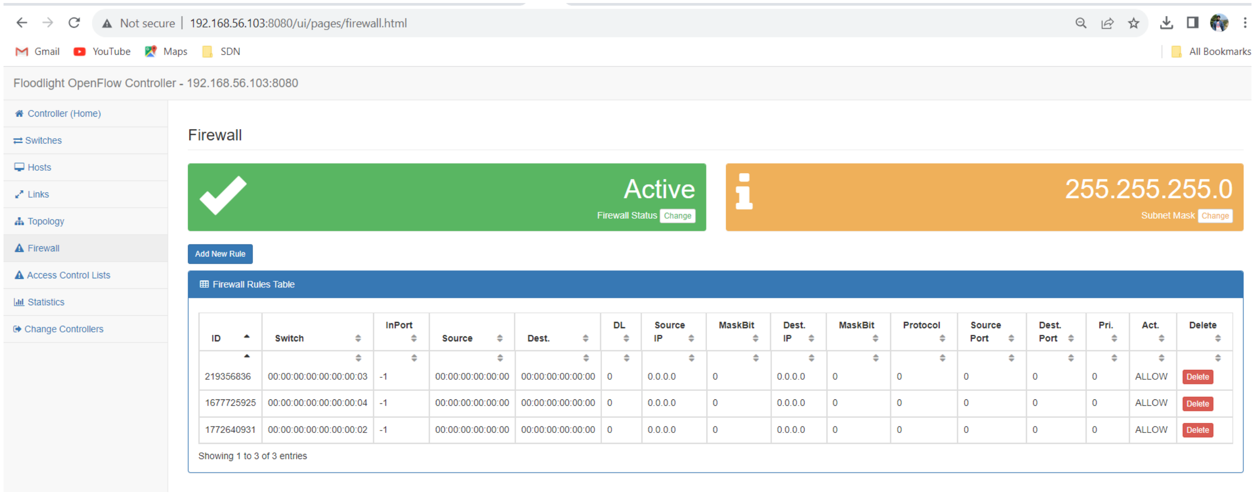
Show 10 entries Search:

Table No	Pkt.Count	Byte	Duration(s)	Priority	IdleTimeoutSec	HardTimeoutSec	Flags	Instructions
0x0	1	42	1	1	5	0		
0x0	1	42	4	1	5	0		
0x0	1	42	4	1	5	0		
0x0	1	42	0	1	5	0		
0x0	1	42	4	1	5	0		
0x0	1	42	3	1	5	0		
0x0	2	84	7	1	5	0		
0x0	1	42	5	1	5	0		
0x0	1	42	2	1	5	0		
0x0	1	42	4	1	5	0		

Showing 1 to 10 of 420 entries

Previous 1 2 3 4 5 … 42 Next

6. Initiate any topology from Mininet and create a firewall rule of your choice. Post the screenshot of the firewall setting activated on the GUI. Also post screenshots of the Floodlight console logs and Mininet pings to prove that the firewall setting is working. **[10 points]**



Not secure | 192.168.56.103:8080/ui/pages/firewall.html

Floodlight OpenFlow Controller - 192.168.56.103:8080

Firewall

Active
Firewall Status Change

255.255.255.0
Subnet Mask Change

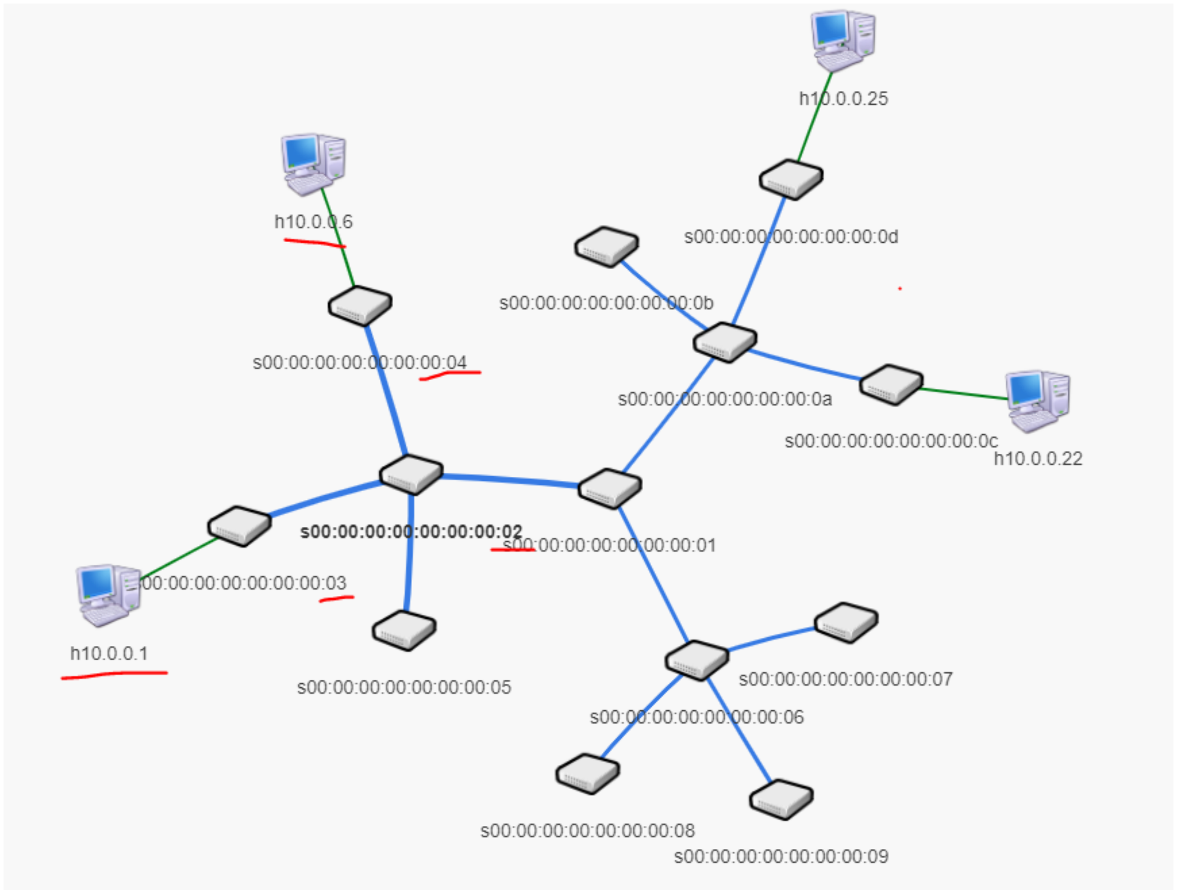
Add New Rule

Firewall Rules Table

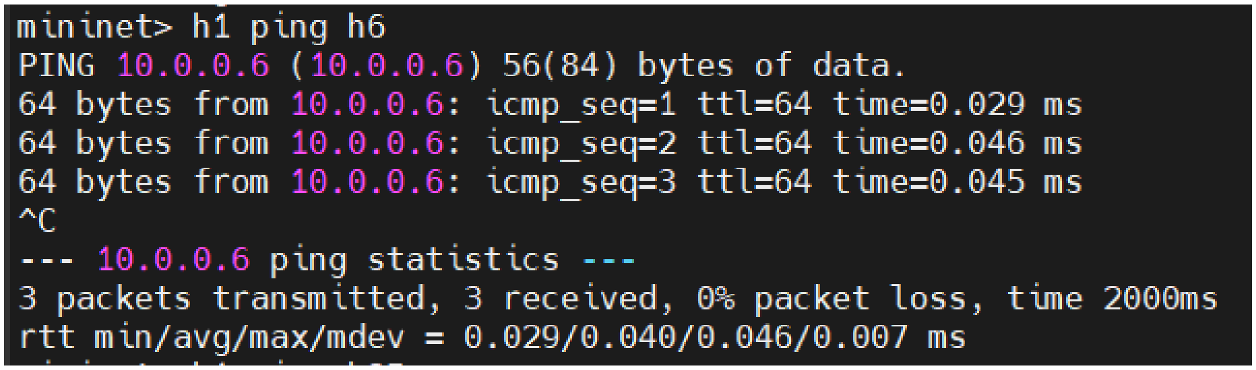
ID	Switch	InPort	Source	Dest.	DL	Source IP	MaskBit	Dest. IP	MaskBit	Protocol	Source Port	Dest. Port	Pri.	Act.	Delete
219356836	00:00:00:00:00:00:00:03	-1	00:00:00:00:00:00	00:00:00:00:00:00	0	0.0.0.0	0	0.0.0.0	0	0	0	0	0	ALLOW	Delete
1677725925	00:00:00:00:00:00:00:04	-1	00:00:00:00:00:00	00:00:00:00:00:00	0	0.0.0.0	0	0.0.0.0	0	0	0	0	0	ALLOW	Delete
1772640931	00:00:00:00:00:00:00:02	-1	00:00:00:00:00:00	00:00:00:00:00:00	0	0.0.0.0	0	0.0.0.0	0	0	0	0	0	ALLOW	Delete

Showing 1 to 3 of 3 entries

The above screenshot, of firewall specifies to allow traffic flowing on switch - S4, S2 and S3

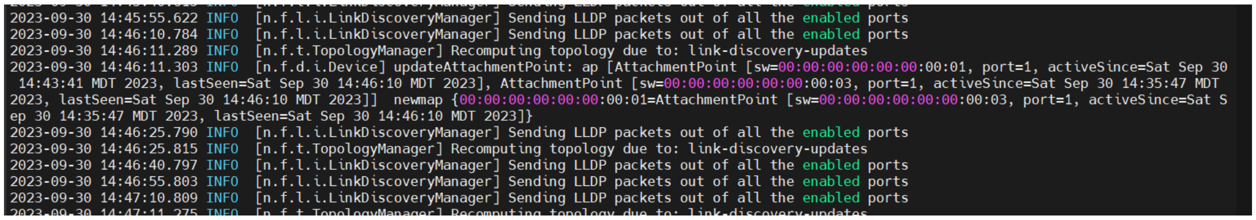


ping from h1 to h6 works!



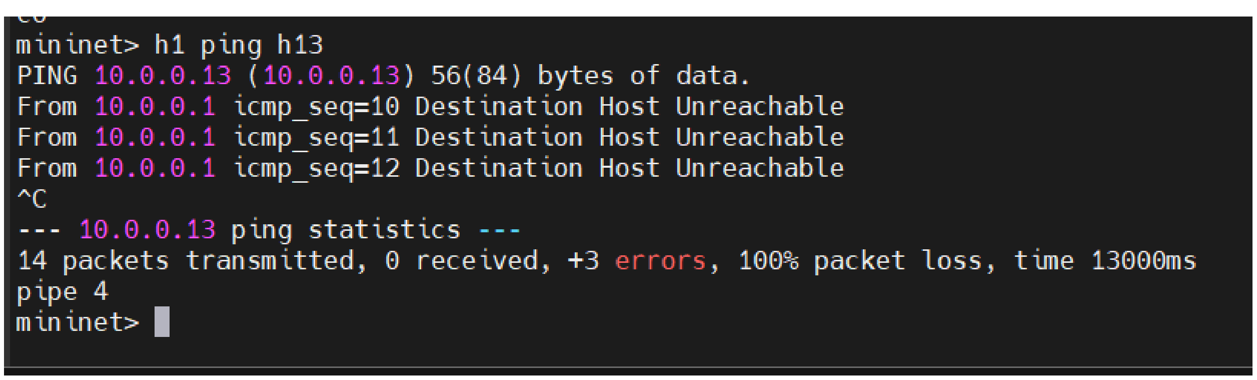
```
mininet> h1 ping h6
PING 10.0.0.6 (10.0.0.6) 56(84) bytes of data.
64 bytes from 10.0.0.6: icmp_seq=1 ttl=64 time=0.029 ms
64 bytes from 10.0.0.6: icmp_seq=2 ttl=64 time=0.046 ms
64 bytes from 10.0.0.6: icmp_seq=3 ttl=64 time=0.045 ms
^C
--- 10.0.0.6 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 0.029/0.040/0.046/0.007 ms
```

Logs -



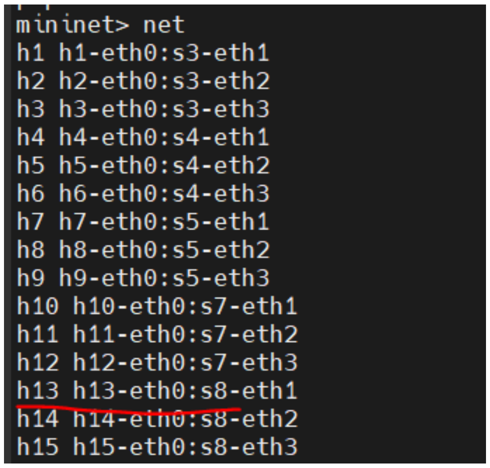
```
2023-09-30 14:45:55.622 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
2023-09-30 14:46:10.784 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
2023-09-30 14:46:11.289 INFO  [n.f.t.TopologyManager] Recomputing topology due to: link-discovery-updates
2023-09-30 14:46:11.303 INFO  [n.f.d.i.Device] updateAttachmentPoint: ap [AttachmentPoint [sw=00:00:00:00:00:00:00:01, port=1, activeSince=Sat Sep 30
 14:43:41 MDT 2023, lastSeen=Sat Sep 30 14:46:10 MDT 2023], AttachmentPoint [sw=00:00:00:00:00:00:00:03, port=1, activeSince=Sat Sep 30 14:35:47 MDT
2023, lastSeen=Sat Sep 30 14:46:10 MDT 2023]]  newmap {00:00:00:00:00:00:00:01=AttachmentPoint [sw=00:00:00:00:00:00:00:03, port=1, activeSince=Sat S
ep 30 14:35:47 MDT 2023, lastSeen=Sat Sep 30 14:46:10 MDT 2023]}
2023-09-30 14:46:25.790 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
2023-09-30 14:46:25.815 INFO  [n.f.t.TopologyManager] Recomputing topology due to: link-discovery-updates
2023-09-30 14:46:40.797 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
2023-09-30 14:46:55.803 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
2023-09-30 14:47:10.809 INFO  [n.f.l.i.LinkDiscoveryManager] Sending LLDP packets out of all the enabled ports
```

ping h1 to h13, should not work



```
mininet> h1 ping h13
PING 10.0.0.13 (10.0.0.13) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=10 Destination Host Unreachable
From 10.0.0.1 icmp_seq=11 Destination Host Unreachable
From 10.0.0.1 icmp_seq=12 Destination Host Unreachable
^C
--- 10.0.0.13 ping statistics ---
14 packets transmitted, 0 received, +3 errors, 100% packet loss, time 13000ms
pipe 4
mininet>
```

We have not allowed any traffic to flow through switch s8, hence h13 would not be reachable



```
mininet> net
h1 h1-eth0:s3-eth1
h2 h2-eth0:s3-eth2
h3 h3-eth0:s3-eth3
h4 h4-eth0:s4-eth1
h5 h5-eth0:s4-eth2
h6 h6-eth0:s4-eth3
h7 h7-eth0:s5-eth1
h8 h8-eth0:s5-eth2
h9 h9-eth0:s5-eth3
h10 h10-eth0:s7-eth1
h11 h11-eth0:s7-eth2
h12 h12-eth0:s7-eth3
h13 h13-eth0:s8-eth1
h14 h14-eth0:s8-eth2
h15 h15-eth0:s8-eth3
```

Objective 5 - Import Custom Topology in Mininet

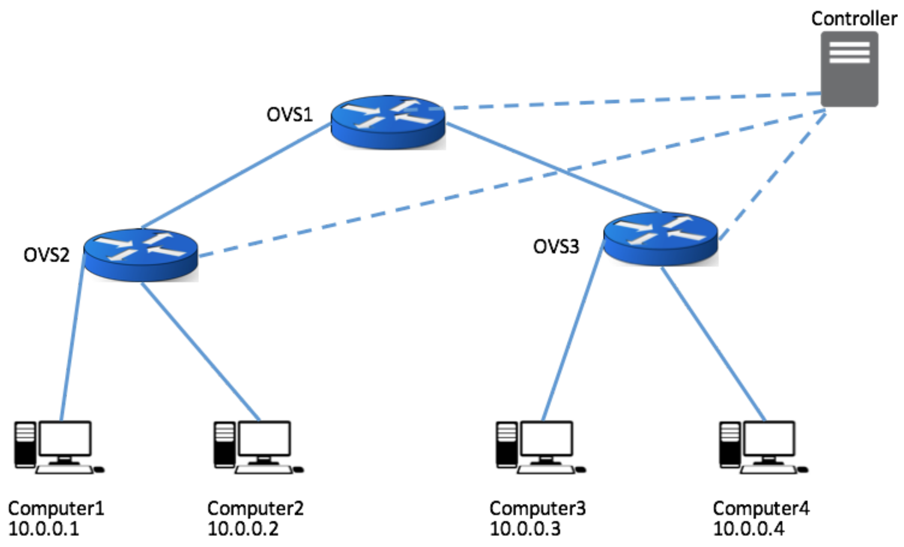
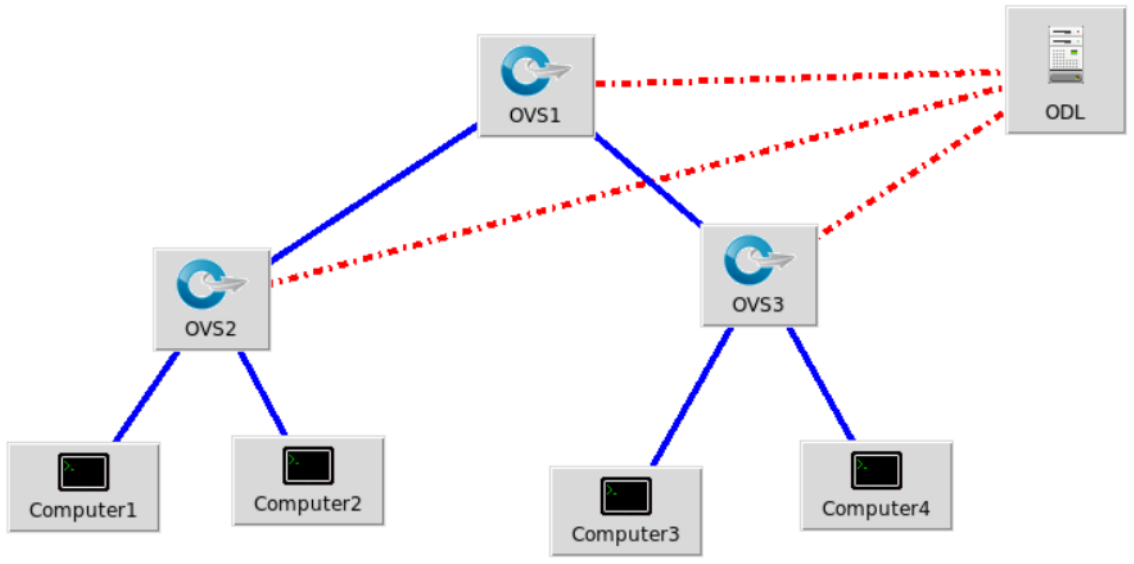
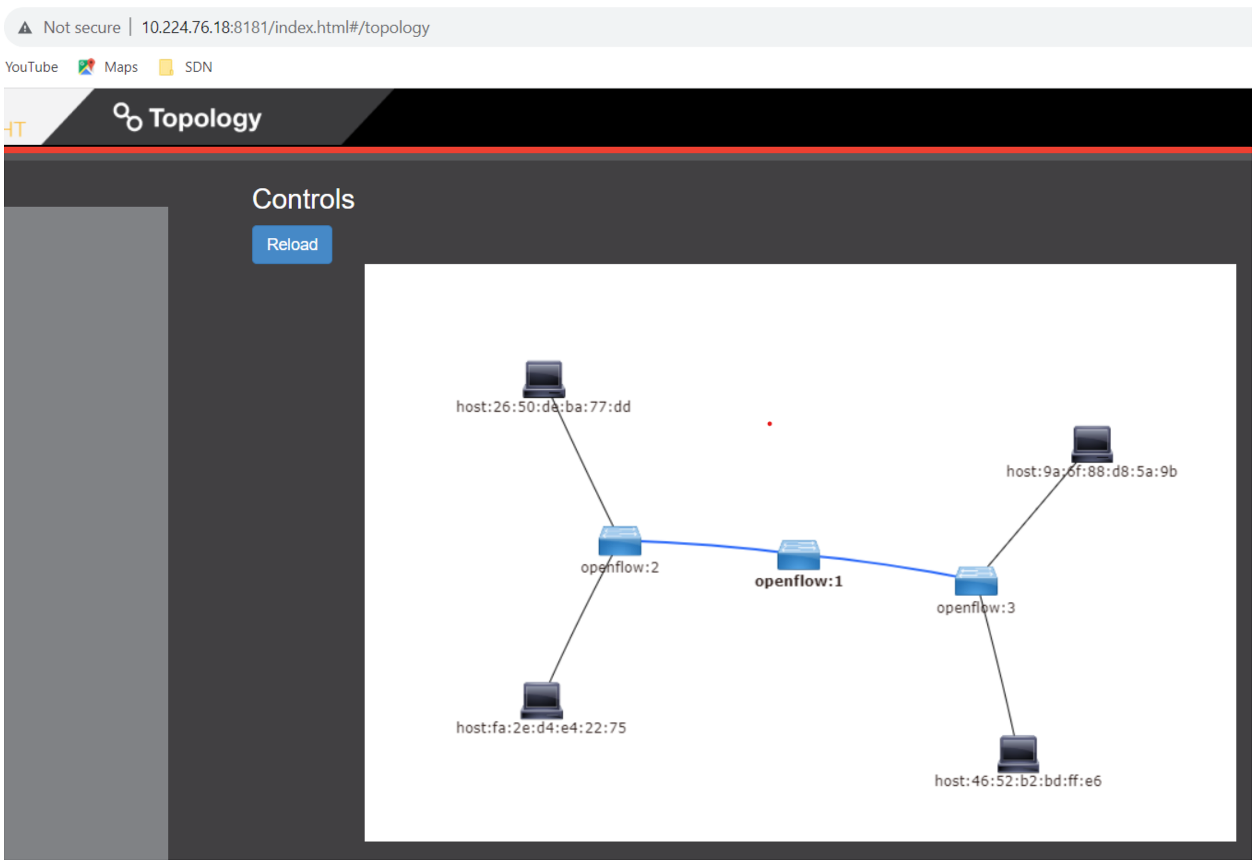


Figure 1. Mininet custom topology

In this objective, you are required to build a custom topology for Mininet as shown in Figure 1 above. All links marked with solid lines need to be set to 10Mbps.

1. Start the controller of your choice, write down the IP of the controller.
2. Build the topology in MiniEdit in your Mininet VM.
    a. Create the topology, set device names, and link speed. Set controller to remote and specify the IP address and port number.
    b. Provide a screenshot of the topology. **[10 points]**

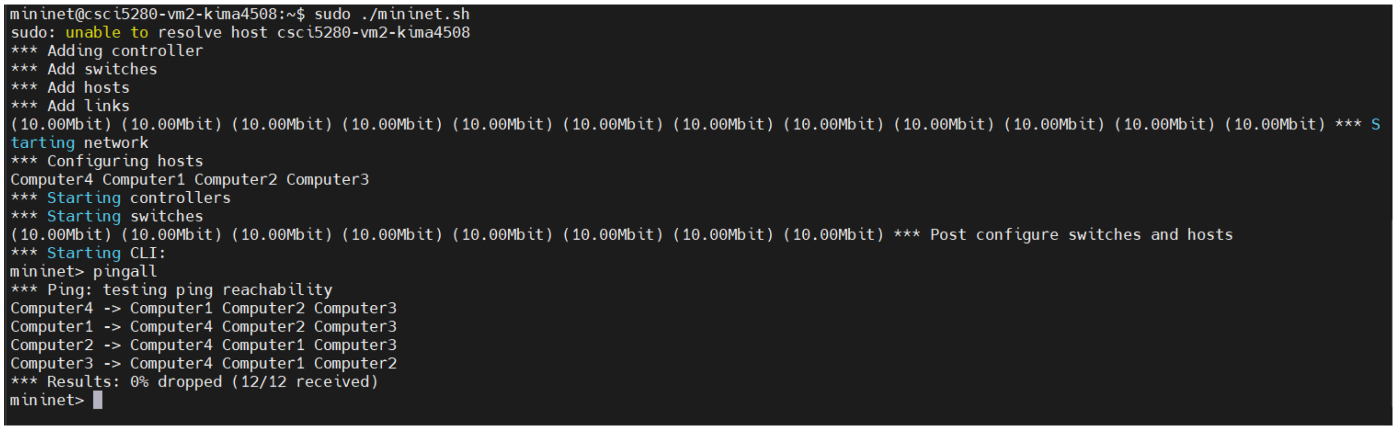

OVS1
ODL
OVS2
OVS3
Computer1
Computer2
Computer3
Computer4


Not secure | 10.224.76.18:8181/index.html#/topology
YouTube
Maps
SDN
Topology
Controls
Reload
host:26:50:de:ba:77:dd
host:9a:6f:88:d8:5a:9b
openflow:2
openflow:1
openflow:3
host:fa:2e:d4:e4:22:75
host:46:52:b2:bd:ff:e6

3. Select “Export Level 2 Script” in the File menu, save the file as .sh file.
4. In your Mininet VM run the following command:

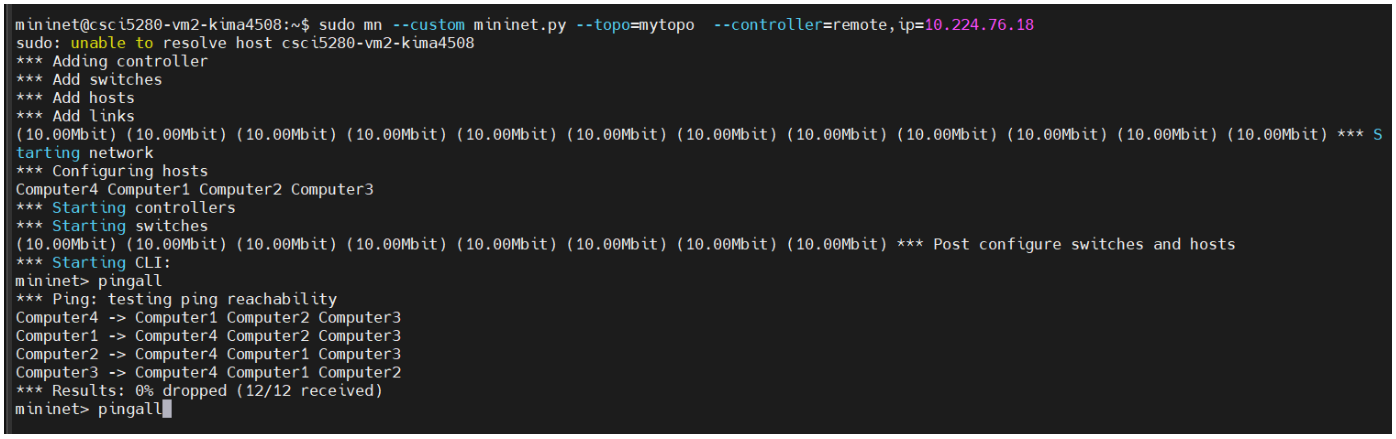
   chmod +x [your .sh file]

   Explain what this command does. **[2 points]**
5. Run the following command to start Mininet
   a. sudo ./[your .sh file]
   b. pingall
   c. Provide a screenshot of the command and Mininet running with results of the pingall **[10 points]**



```
mininet@csci5280-vm2-kima4508:~$ sudo ./mininet.sh
sudo: unable to resolve host csci5280-vm2-kima4508
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
(10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) *** S
tarting network
*** Configuring hosts
Computer4 Computer1 Computer2 Computer3
*** Starting controllers
*** Starting switches
(10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) *** Post configure switches and hosts
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
Computer4 -> Computer1 Computer2 Computer3
Computer1 -> Computer4 Computer2 Computer3
Computer2 -> Computer4 Computer1 Computer3
Computer3 -> Computer4 Computer1 Computer2
*** Results: 0% dropped (12/12 received)
mininet> 
```

6. Now you loaded Mininet with a custom topology using a .sh file. There is another way to load a custom topology, by using the following command:
   a. sudo mn --custom topology.py --topo=mytopo --controller=remote,ip=xx.xx.xx.xx
   b. pingall
7. Modify the .sh file into .py file that can be loaded in the above command. More details can be found here http://mininet.org/walkthrough/#custom-topologies and here http://mininet.org/api/classmininet_1_1topo_1_1Topo.html.
   a. Provide a screenshot of the command and Mininet running with results of the “pingall” command. **[10 points]**



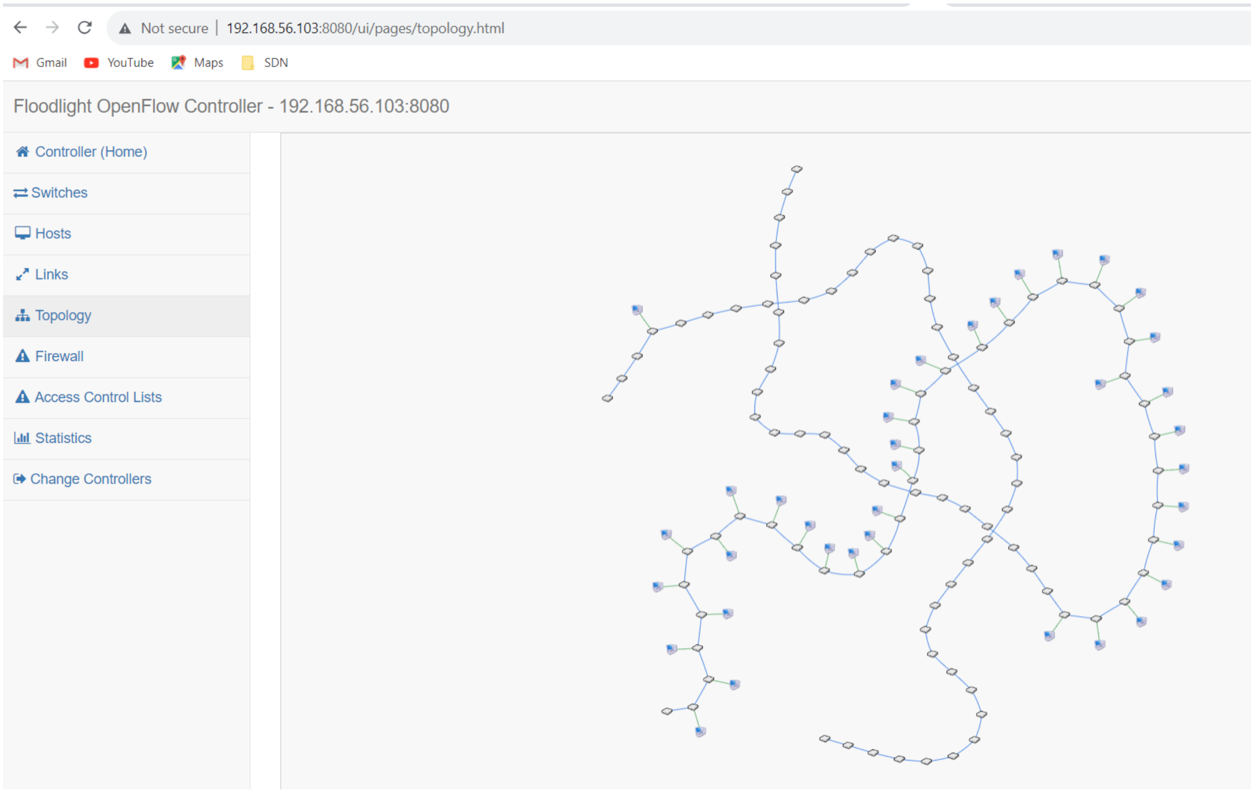
```
mininet@csci5280-vm2-kima4508:~$ sudo mn --custom mininet.py --topo=mytopo  --controller=remote,ip=10.224.76.18
sudo: unable to resolve host csci5280-vm2-kima4508
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
(10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) *** S
tarting network
*** Configuring hosts
Computer4 Computer1 Computer2 Computer3
*** Starting controllers
*** Starting switches
(10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) *** Post configure switches and hosts
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
Computer4 -> Computer1 Computer2 Computer3
Computer1 -> Computer4 Computer2 Computer3
Computer2 -> Computer4 Computer1 Computer3
Computer3 -> Computer4 Computer1 Computer2
*** Results: 0% dropped (12/12 received)
mininet> pingall
```

8. Provide original .sh file, modified .sh file, and the .py file with your report to get points.

Objective 6 - Real World

1. Use any controller you like.
2. Create a huge topology of a multitude of switches or hosts or both (be careful not to crash your PC!).
    a. Provide a screenshot of the nightmare topology from the controller GUI. **[1 points]**

       mininet@mininet-ofm:~$ sudo mn --controller=remote,ip=192.168.56.103 --switch ovsk,protocols=OpenFlow13 --topo=linear,100 --mac



b. Explain why this topology could possibly be realistic of a real world SDN network design. **[5 points]**

It's highly likely that an ISP can have 100 switches connected in a linear fashion as a part of their huge topology, where the switches can be geographically distributed across the globe/continents. This would mean a controller to have connectivity to each switch and should have a holistic view to make routing based on its tenants.

Objective 7 – Report questions

1. After completing this lab, which controller would you prefer and why? [**5 points**]

   ONOS is more preferred as the community support is wide and the controller is more efficient in terms of implementation and what it offers.
   Intent based networking could help achieve a lot of objectives at a push of a button. Although it is complex due to its functionality, it abstracts it to the user making it a better choice.

2. How can we make a controller communicate with a traditional router/switch not running

   OpenFlow for configuration/monitoring purposes? [**5 points**]

   An ISP can talk to another ISP. It is likely that a BGP connection needs to be established with another provider's edge router. In such scenarios, a BGP app needs to run in the Application layer which would process the packet in message and generate a packet out message every x seconds for the neighbourship to be up.
   A keep-alive message received from the traditional switch to a Openflow switch would be sent to the controller as a packet-in message. The BGP App would see this message and keep the neighbourship in its database. Also, the BGP App would send periodic messages to be sent out via the Openflow switch to be sent towards the traditional router. In simple terms, the neighbourship is formed between the BGP App in the Openflow domain and the traditional router.

Total Score ______________ / 320