AUM Amriteswaryai Namah

Assignment 2 on Mininet

Name: J Viswaksena Roll.No: AM.EN.UAIE21035

Deliverable

1. Screenshot for Pingall before and after adding flow table entries.
2. Screenshot for dump-flow commands

1. Run this command to create a simple host with 3 hosts and 1 Open vswitch and a remote controller sudo mn --topo single,3 --mac --switch ovsk --controller remote

```
Viswaksena@ubuntu:~$ sudo mn --topo single,3 --mac --switch ovsk --controller remot
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

2. Run a net command and see all the interfaces and find out how the switches are connected to each other.

```
mininet> net info
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0
c0
```

3. Run a pingall command on mininet prompt and write the output. Explain the reason why you got the output that way.

   Due to absence of entries in flow table, packets are struct which host it need to reach. So for this reason 100% packet loss get occurs. To receive packets we need to give inputs to the flow table then they can do conversation.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X X
h3 -> X X
*** Results: 100% dropped (0/6 received)
```

4. Next, you see the flow table at the switch s1

```
mininet> dpctl show
*** s1 ------------------------------------------------------------------------
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000001
n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src mod_dl_dst mod_
nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst
 1(s1-eth1): addr:0e:d6:bf:22:87:15
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 2(s1-eth2): addr:e6:db:4c:05:87:39
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 3(s1-eth3): addr:52:b0:be:15:a7:7b
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 LOCAL(s1): addr:c2:a2:34:9d:f7:48
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

5. $ovs-ofctl add-flow tcp:127.0.0.1:6654 in_port=1,idle_timeout=1000,actions=output:2

```
Viswaksena@ubuntu:~$ ovs-ofctl add-flow tcp:127.0.0.1:6654 in_port=1,idle_timeo
ut=1000,actions=output:2
```

6. $ovs-ofctl add-flow tcp:127.0.0.1:6654 in_port=2,idle_timeout=1000,actions=output:1

```
Viswaksena@ubuntu:~$ ovs-ofctl add-flow tcp:127.0.0.1:6654 in_port=2,idle_timeo
ut=1000,actions=output:1
```

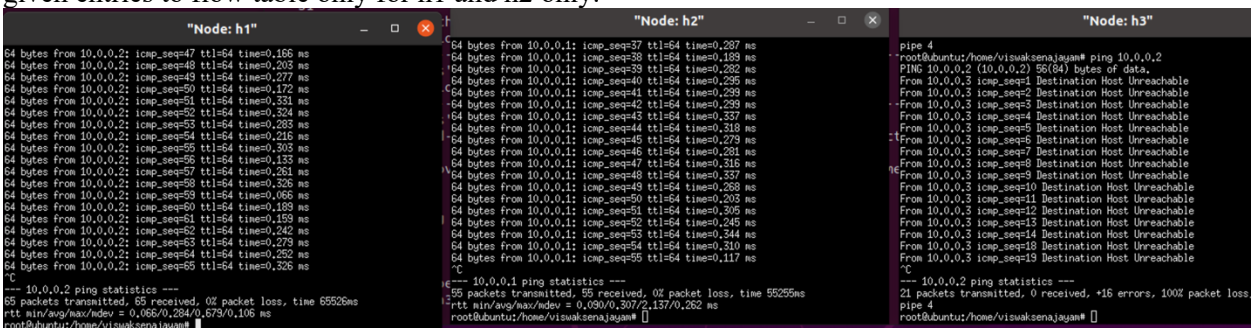
7. $ovs-ofctl dump-flows tcp:127.0.0.1:6654

```
Viswaksena@ubuntu:~$ ovs-ofctl dump-flows tcp:127.0.0.1:6654
 cookie=0x0, duration=30.939s, table=0, n_packets=0, n_bytes=0, idle_timeout=10
00, in_port="s1-eth1" actions=output:"s1-eth2"
 cookie=0x0, duration=9.838s, table=0, n_packets=0, n_bytes=0, idle_timeout=100
0, in_port="s1-eth2" actions=output:"s1-eth1"
```

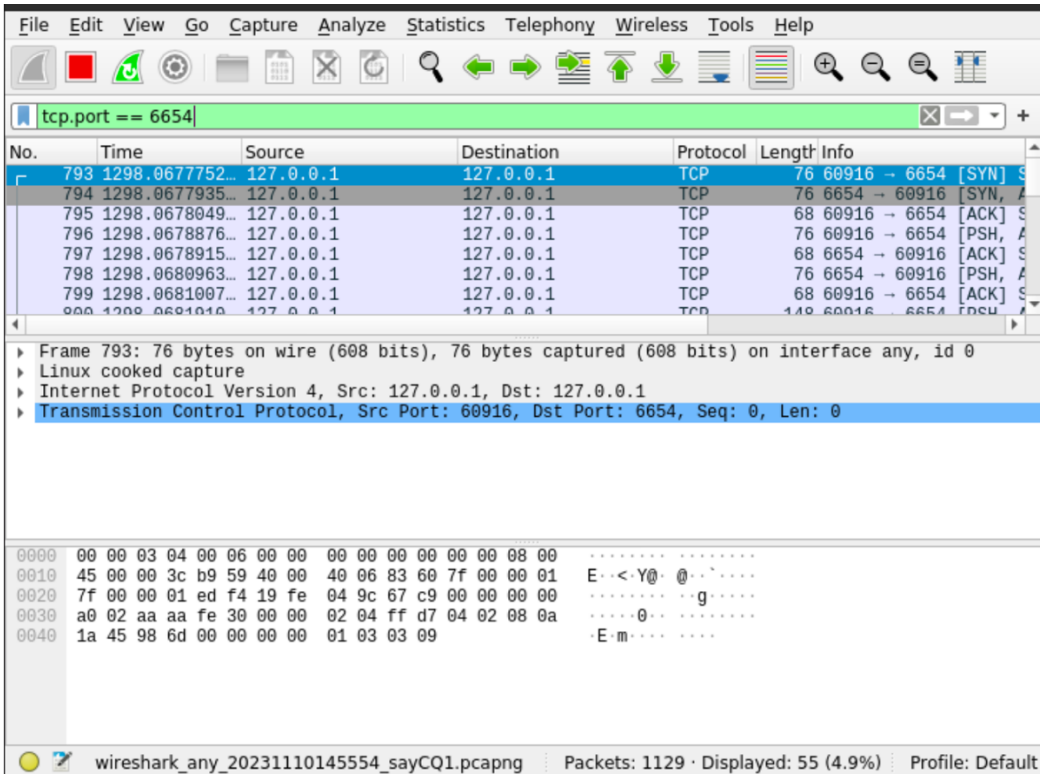
8. Mininet> pingall

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X
h2 -> h1 X
h3 -> X X
*** Results: 66% dropped (2/6 received)
```

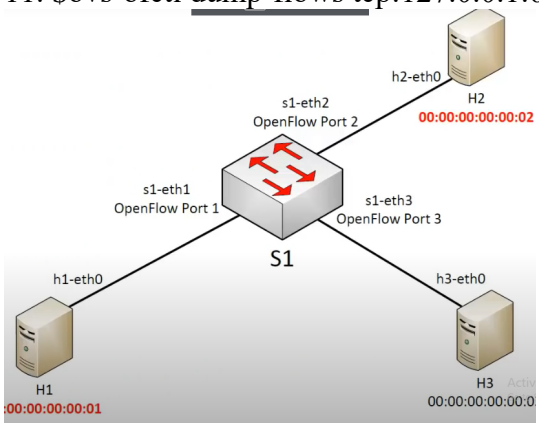
9. **h1 and h2 are now connected. [If there are switches, Port 6655 is used for Switch S2]**
   From the below diagram we can say that the host 1 and host 2 are connected transmitted packets from h1 to h2, we get 0% packet loss, same from h2 to h1. But from h3 to h2,h1 transmitted 21 packets then happens 100% packet loss. h3 to h1 either h2 are not connected. Because we only given entries to flow table only for h1 and h2 only.



10. Start Wireshark on a window and before doing the next dump-flows command, start capturing packets.



11. $ovs-ofctl dump-flows tcp:127.0.0.1:6654 and check the statistics on packet sent



After adding links

```
Viswaksena@ubuntu:~$ ovs-ofctl dump-flows tcp:127.0.0.1:6654
 cookie=0x0, duration=1940.489s, table=0, n_packets=4, n_bytes=280, idle_timeout
=1000, in_port="s1-eth1" actions=output:"s1-eth2"
 cookie=0x0, duration=1914.558s, table=0, n_packets=4, n_bytes=280, idle_timeout
=1000, in_port="s1-eth2" actions=output:"s1-eth1"
```

*any

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port == 6654

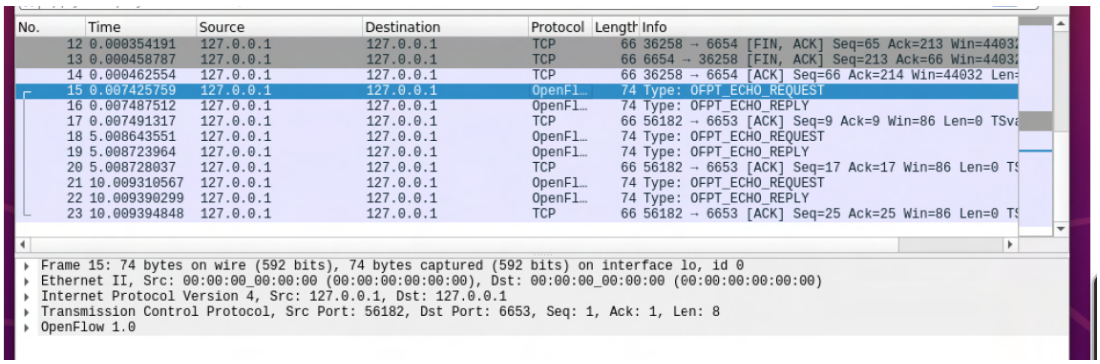
No.	Time	Source	Destination	Protocol	Length	Info
1012	1661.9139042…	127.0.0.1	127.0.0.1	TCP	68	6654 → 55830 [ACK] S
1013	1661.9139987…	127.0.0.1	127.0.0.1	TCP	76	6654 → 55830 [PSH, A
1014	1661.9140032…	127.0.0.1	127.0.0.1	TCP	68	55830 → 6654 [ACK] S
1015	1661.9141480…	127.0.0.1	127.0.0.1	TCP	156	55830 → 6654 [PSH, A
1016	1661.9141972…	127.0.0.1	127.0.0.1	TCP	68	55830 → 6654 [FIN, A
1017	1661.9142497…	127.0.0.1	127.0.0.1	TCP	68	6654 → 55830 [FIN, A
1018	1661.9142554…	127.0.0.1	127.0.0.1	TCP	68	55830 → 6654 [ACK] S

```
▸ Frame 1018: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
▸ Transmission Control Protocol, Src Port: 55830, Dst Port: 6654, Seq: 98, Ack: 10, Len: 0
```

```
0000  00 00 03 04 00 06 00 00  00 00 00 00 00 00 08 00   ........ ........
0010  45 00 00 34 af ae 40 00  40 06 8d 13 7f 00 00 01   E··4··@· @·......
0020  7f 00 00 01 da 16 19 fe  71 98 d4 df 57 b8 3c 8f   ........ q···W·<·
0030  80 10 00 56 fe 28 00 00  01 01 08 0a 1a 4b 25 b3   ···V·(·· ·····K%·
0040  1a 4b 25 b3                                       ·K%·
```

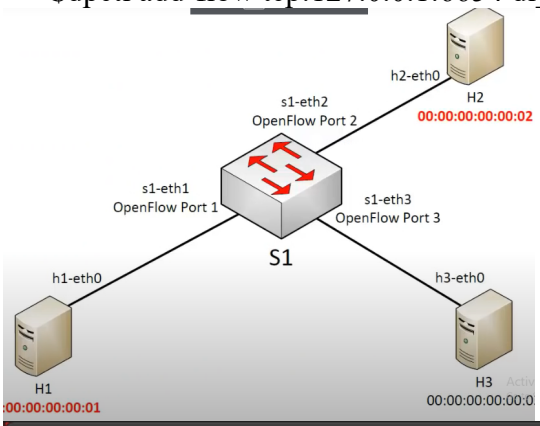
wireshark_any_20231110145554_sayCQ1.pcapng Packets: 1151 · Displayed: 55 (4.8%) Profile: Default

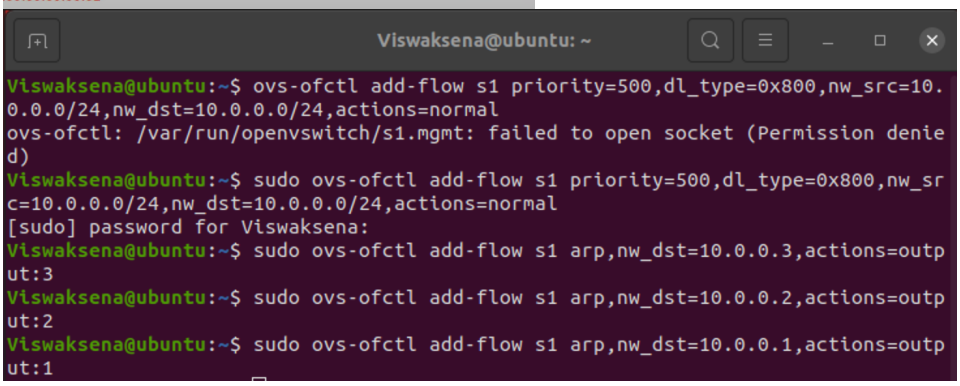
12. Run Wireshark on another window and Identify all the messages corresponding to **OpenFlow protocol**. Take a screenshot and mark two messages – one from switch to Controller and another one Controller to switch
    Sample is shown here

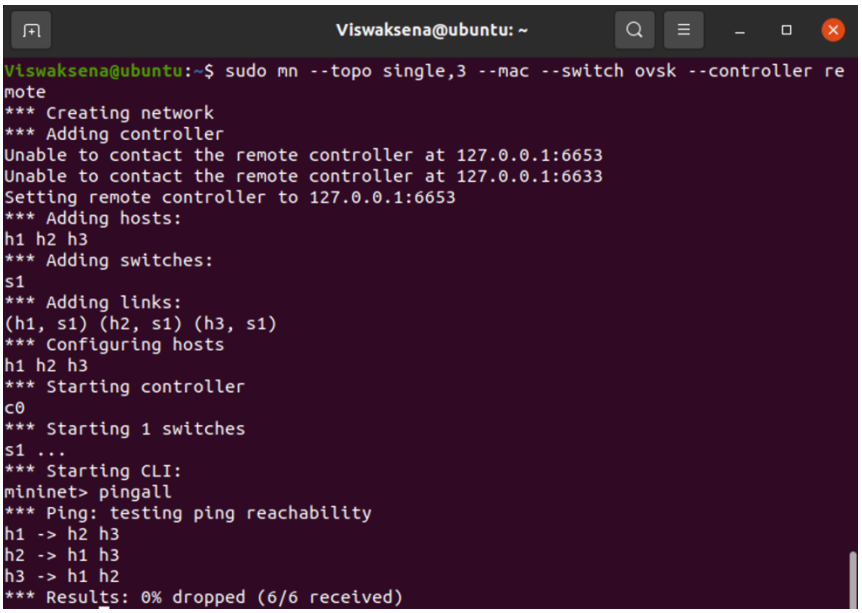


13. Continue the exercise to completely install flow table for all hosts.
14. Ping every pair of hosts and make sure they work. You may have to add the following for the switch which directs packets destined for a host to a particular port. It does it based on MAC address and it cannot be done using ip_src or ip_dst only. In the below commands, dl_src and dl_dst give the source MAC address and destination MAC address

$dpctl add-flow tcp:127.0.0.1:6654 dl_dst=0:0:0:0:0:1,idle_timeout=1000,actions=output:1
$dpctl add-flow tcp:127.0.0.1:6654 dl_dst=0:0:0:0:0:2,idle_timeout=1000,actions=output:2
$dpctl add-flow tcp:127.0.0.1:6654 dl_dst=0:0:0:0:0:3,idle_timeout=1000,actions=output:3

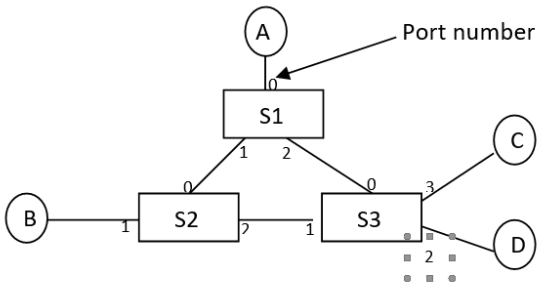






Additional Exercise for Assignment 2 – for those who have already completed.

Assuming that you are managing a network with the following topology where A can be the Controller. The Interface numbers can be different. By default, eth0 of S1 is connected to the controller.
Give -- mac option for invoking the mininet. You can assign specific IP address to a host through Python API.
Please see the examples in the examples folder under your mininet installation directory. For me it is /usr/lib/python2.7/dist-packages/mininet



Assume IP address of A, B, C, D: 10.0.0.1, 10.0.0.2, 10.0.0.3,10.0.0.4

Specify the Openflow rules on switches S1, S2, S3 that enforce the following policies. Make sure that all needed communication including acknowledgement can happen.

- Nodes A, B, and C can talk to one another freely.
- Node D has access to ports 22 and 80 of Nodes A and B, but nothing else.

Node D and Node C cannot talk to each other.

```
Viswaksena@ubuntu:~$ sudo mn --custom /home/viswaksenajayam/mininet_topology.py
 --topo mytopo
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s3) (s1, s2) (s1, s3) (s2, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s2-eth1 (OK OK)
h3-eth0<->s3-eth1 (OK OK)
h4-eth0<->s3-eth2 (OK OK)
s1-eth2<->s2-eth2 (OK OK)
s1-eth3<->s3-eth3 (OK OK)
s2-eth3<->s3-eth4 (OK OK)
```

Code:-

Open

```
from mininet.topo import Topo

class MyTopo(Topo):
    def __init__(self):
        Topo.__init__(self)

        s1 = self.addSwitch('s1')
        s2 = self.addSwitch('s2')
        s3 = self.addSwitch('s3')

        h1 = self.addHost('h1')
        h2 = self.addHost('h2')
        h3 = self.addHost('h3')
        h4 = self.addHost('h4')

        self.addLink(h1, s1)
        self.addLink(h2, s2)
        self.addLink(h3, s3)
        self.addLink(h4, s3)

        self.addLink(s1, s2)
        self.addLink(s1, s3)
        self.addLink(s2, s3)


topos = {'mytopo': (lambda: MyTopo())}
```