Lab 3 - Build a Learning Switch on Ryu

Experiment demo

Step 1

After you install ryu, you could run your code by commad

“ryu-manager yourapp.py”

Ex:

```
shiny@ubuntu:~$ ryu-manager learning_switch.py
loading app learning_switch.py
loading app ryu.controller.ofp_handler
instantiating app learning_switch.py of SimpleSwitch
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Step 2

Run mininet with remote controller(RYU) by commad

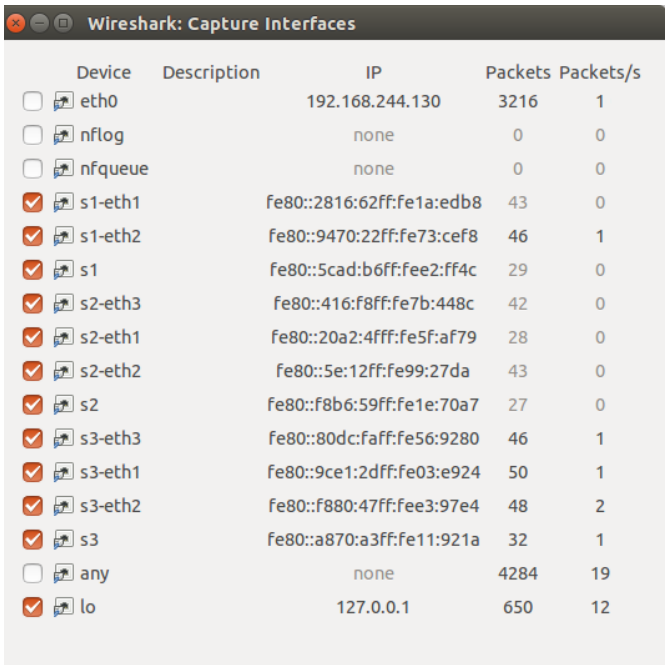
sudo mn --controller=remote --topo tree,2 --mac

Ex:

```
shiny@ubuntu:~$ sudo mn --controller=remote --topo tree,2 --mac
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

Step 3:

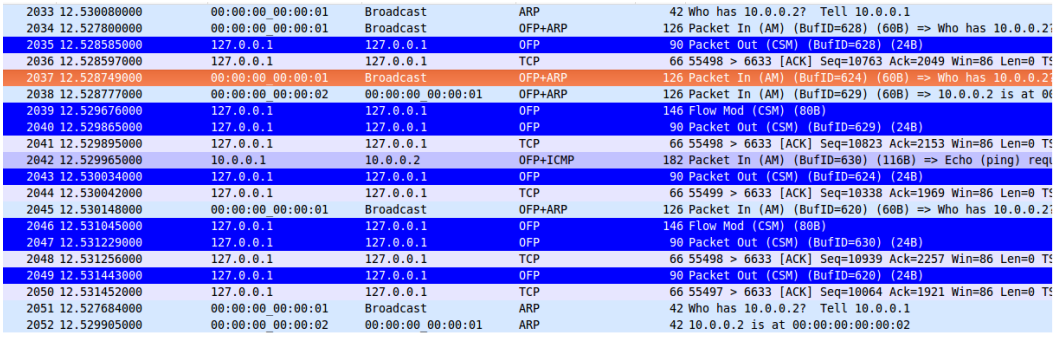
Running your wireshark to capture the packets



Step 4:

Hosts ping each other

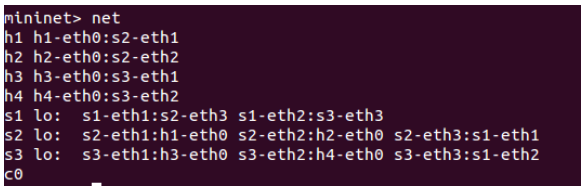
h1 ping h2



2033	12.530080000	00:00:00_00:00:01	Broadcast	ARP	42 Who has 10.0.0.2? Tell 10.0.0.1
2034	12.527800000	00:00:00_00:00:01	Broadcast	OFP+ARP	126 Packet In (AM) (BufID=628) (60B) => Who has 10.0.0.2
2035	12.528585000	127.0.0.1	127.0.0.1	OFP	90 Packet Out (CSM) (BufID=628) (24B)
2036	12.528597000	127.0.0.1	127.0.0.1	TCP	66 55498 > 6633 [ACK] Seq=10763 Ack=2049 Win=86 Len=0 TS
2037	12.528749000	00:00:00_00:00:01	Broadcast	OFP+ARP	126 Packet In (AM) (BufID=624) (60B) => Who has 10.0.0.2
2038	12.528777000	00:00:00_00:00:02	00:00:00_00:00:01	OFP+ARP	126 Packet In (AM) (BufID=629) (60B) => 10.0.0.2 is at 0
2039	12.529676000	127.0.0.1	127.0.0.1	OFP	146 Flow Mod (CSM) (80B)
2040	12.529865000	127.0.0.1	127.0.0.1	OFP	90 Packet Out (CSM) (BufID=629) (24B)
2041	12.529895000	127.0.0.1	127.0.0.1	TCP	66 55498 > 6633 [ACK] Seq=10823 Ack=2153 Win=86 Len=0 TS
2042	12.529965000	10.0.0.1	10.0.0.2	OFP+ICMP	182 Packet In (AM) (BufID=630) (116B) => Echo (ping) requ
2043	12.530034000	127.0.0.1	127.0.0.1	OFP	90 Packet Out (CSM) (BufID=624) (24B)
2044	12.530042000	127.0.0.1	127.0.0.1	TCP	66 55499 > 6633 [ACK] Seq=10338 Ack=1969 Win=86 Len=0 TS
2045	12.530148000	00:00:00_00:00:01	Broadcast	OFP+ARP	126 Packet In (AM) (BufID=620) (60B) => Who has 10.0.0.2
2046	12.531045000	127.0.0.1	127.0.0.1	OFP	146 Flow Mod (CSM) (80B)
2047	12.531229000	127.0.0.1	127.0.0.1	OFP	90 Packet Out (CSM) (BufID=630) (24B)
2048	12.531256000	127.0.0.1	127.0.0.1	TCP	66 55498 > 6633 [ACK] Seq=10939 Ack=2257 Win=86 Len=0 TS
2049	12.531443000	127.0.0.1	127.0.0.1	OFP	90 Packet Out (CSM) (BufID=620) (24B)
2050	12.531452000	127.0.0.1	127.0.0.1	TCP	66 55497 > 6633 [ACK] Seq=10064 Ack=1921 Win=86 Len=0 TS
2051	12.527684000	00:00:00_00:00:01	Broadcast	ARP	42 Who has 10.0.0.2? Tell 10.0.0.1
2052	12.529905000	00:00:00_00:00:02	00:00:00_00:00:01	ARP	42 10.0.0.2 is at 00:00:00:00:00:02

Step 5

Check the flow entry on SDN switch by command



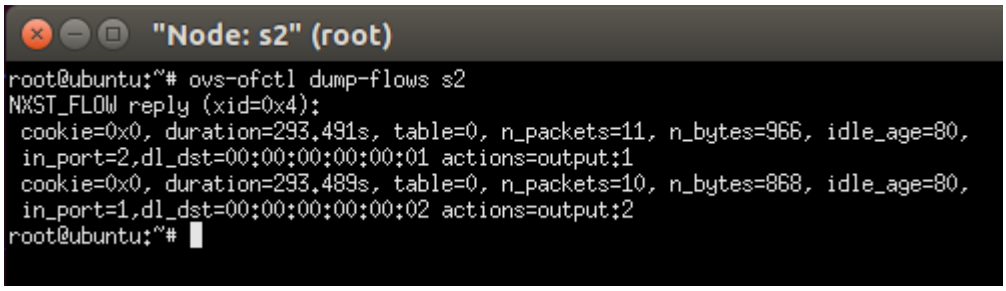
```
mininet> net
h1 h1-eth0:s2-eth1
h2 h2-eth0:s2-eth2
h3 h3-eth0:s3-eth1
h4 h4-eth0:s3-eth2
s1 lo:  s1-eth1:s2-eth3 s1-eth2:s3-eth3
s2 lo:  s2-eth1:h1-eth0 s2-eth2:h2-eth0 s2-eth3:s1-eth1
s3 lo:  s3-eth1:h3-eth0 s3-eth2:h4-eth0 s3-eth3:s1-eth2
c0
```

- xterm s2 (depend on the switch you use), here h1 is connecting with s2
- ovs-ofctl dump-flows s2 (depend on the switch you use)

And then you suppose see the flow entry(rules) on switch

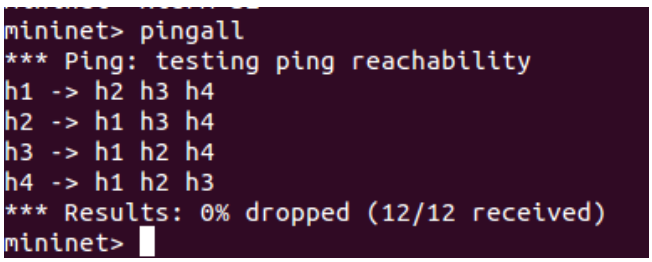
Which means in_port = 1, mac_dst=00:00:00:00:00:02 => action=output to port2

(means that the packets coming from h1(in_port = 1) and dst=h2 (mac_dst=00:00:00:00:00:02), packets will follow the rule output to port 2 which h2 connected)



```
"Node: s2" (root)
root@ubuntu:~# ovs-ofctl dump-flows s2
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=293.491s, table=0, n_packets=11, n_bytes=966, idle_age=80,
 in_port=2,dl_dst=00:00:00:00:00:01 actions=output:1
 cookie=0x0, duration=293.489s, table=0, n_packets=10, n_bytes=868, idle_age=80,
 in_port=1,dl_dst=00:00:00:00:00:02 actions=output:2
root@ubuntu:~#
```

Finally, you could use command “pingall” to test if your code works properly.



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

Step 6

Any plagiarism is prohibited (including codes and report)

Summit your code and report of ***how your code design*** and **observations** whatever you learn (ex: openflow protocol) during the experiment.

Some info you will use:

- datapath.ofproto_parser.OFPActionOutput(*port*, *max_len=65509*, *type_=None*, *len_=None*)

Output action

This action indicates output a packet to the switch port.

Attribute	**Description**
port	Output port
max_len	Max length to send to controller

- out_port = ofproto.OFPP_FLOOD

when you want to flood the packet, set the port = ofproto.OFPP_FLOOD

- You will need to define a dictionary to store the mapping relations between mac address and incoming port