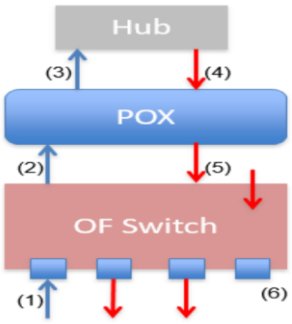
PRACTICAL-5

AIM:

Implement the basic hub example using Pox controller and verify Hub behaviour with tcpdump. Also, Create the learning switch.



THEORY:

POX CONTROLLER:

- POX is an open source development platform for Python-based software-defined networking (SDN) control applications.
- It creates a realistic virtual network, running real kernel, switch and application code on a single machine.
- It adds a listener that listen to openflow switches for connection.

TCP DUMP:

- TCP Dump is a widely used command line packet analyzer tool.
- It is used to capture and filter tcp/ip packets that are received or transfer over a network on a Specific interface.

IMPLEMENTATION:

- First, we will open the terminal and navigate to pox folder.

```
parth642001@ubuntu:~$ cd pox
parth642001@ubuntu:~/pox$ ./pox.py forwarding.hub
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley,
INFO:forwarding.hub:Proactive hub running.
WARNING:version:Support for Python 3 is experimental.
INFO:core:POX 0.7.0 (gar) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
INFO:forwarding.hub:Hubifying 00-00-00-00-00-01
```

Starting the POX controller

- Now, open another terminal and create the topology

```
parth642001@ubuntu:~$ sudo mn --topo single,3 --mac --switch ovsk --controller r
emote
[sudo] password for parth642001:
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** 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:
```

Topology created

- Once the topology is created, controller will detect and will connect to the hub.

```
INFO:core:POX 0.7.0 (gar) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
INFO:forwarding.hub:Hubifying 00-00-00-00-00-01
```

POX controller connected to hub

- Now, we will check the connection

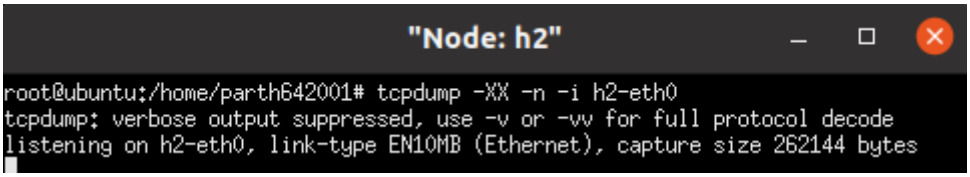
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

Performing pingall

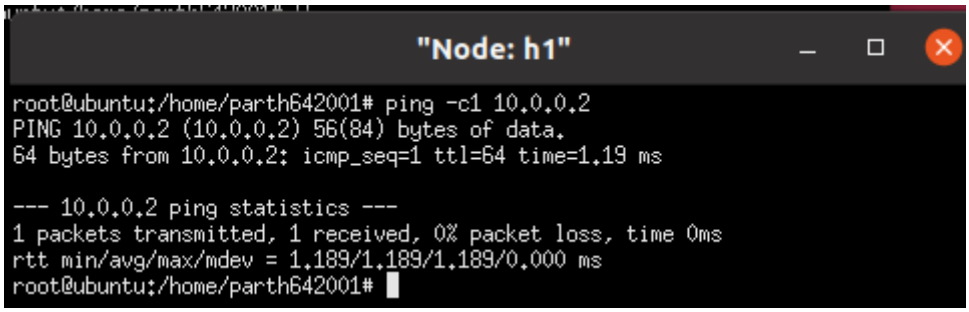
```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
 cookie=0x0, duration=1367.333s, table=0, n_packets=68, n_bytes=5204, actions=FL
OOD
```

Flow table entry

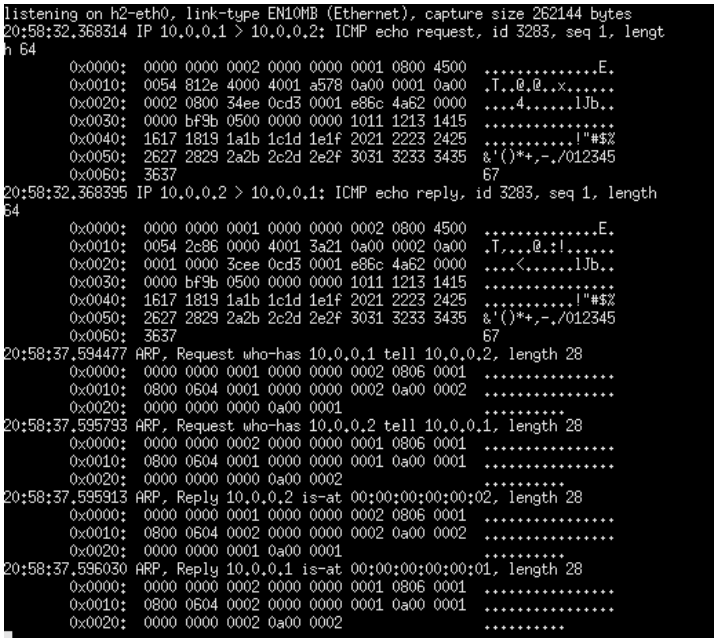
- Now, we will verify the hub's behaviour with tcpdump
- We will give the command to open console of individual mode.



We will set up h2 to receive packets

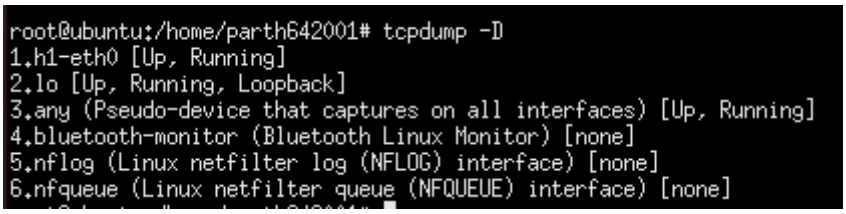


Sending packets from h1 to h2

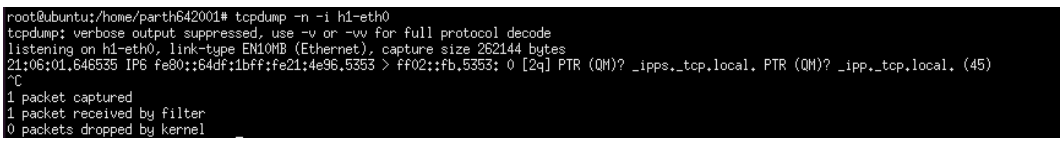


H2 receiving the packets

- We will test certain commands
- Command: tcpdump -D
- It displays the available interfaces



- Command: tcpdump -n -i eth0
- It captures the ip packets



- Now, we will create a learning switch.
- We will open the terminal and we will navigate to pox folder to start the controller learning switch.

```
parth642001@ubuntu:~$ cd pox
parth642001@ubuntu:~/pox$ ./pox.py forwarding.l2_learning
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
WARNING:version:Support for Python 3 is experimental.
INFO:core:POX 0.7.0 (gar) is up.
```

- Now, we will open another terminal to create topology

```
parth642001@ubuntu:~$ sudo mn --topo single,3 --mac --switch ovsk --controller r
emote
[sudo] password for parth642001:
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** 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:
```

Topology created

- Also, controller has connected to the switch

```
INFO:core:POX 0.7.0 (gar) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
```

CONCLUSION:

- In this practical, I learnt the basics of POX controller and the working of tcp-dump