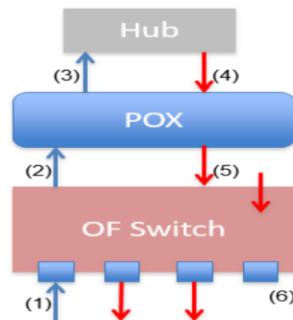


PRACTICAL-5

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 remote
[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.

```
"Node: h2"
root@ubuntu:/home/parth642001# tcpdump -XX -n -i h2-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h2-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

We will set up h2 to receive packets

```
"Node: h1"
root@ubuntu:/home/parth642001# ping -c1 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.19 ms

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.189/1.189/1.189/0.000 ms
root@ubuntu:/home/parth642001#
```

Sending packets from h1 to h2

```

listening on h2-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
20:58:32.368314 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 3283, seq 1, length 64
0x0000: 0000 0000 0002 0000 0000 0001 0800 4500 .....E.
0x0010: 0054 812e 4000 4001 a578 0a00 0001 0a00 .T...@.x.....
0x0020: 0002 0800 34ee 0cd3 0001 e86c 4a62 0000 ....4.....lJb..
0x0030: 0000 bf9b 0500 0000 0000 1011 1213 1415 .....
0x0040: 1617 1819 1a1b 1c1d 1e1f 2021 2223 2425 .....!""#$%
0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 &'()*+,-./012345
0x0060: 3637 .....67
20:58:32.368395 IP 10.0.0.2 > 10.0.0.1: ICMP echo reply, id 3283, seq 1, length 64
0x0000: 0000 0000 0001 0000 0000 0002 0800 4500 .....E.
0x0010: 0054 2c86 0000 4001 3a21 0a00 0002 0a00 .T...@.:!.....
0x0020: 0001 0000 3cee 0cd3 0001 e86c 4a62 0000 ....<.....lJb..
0x0030: 0000 bf9b 0500 0000 0000 1011 1213 1415 .....
0x0040: 1617 1819 1a1b 1c1d 1e1f 2021 2223 2425 .....!""#$%
0x0050: 2627 2829 2a2b 2c2d 2e2f 3031 3233 3435 &'()*+,-./012345
0x0060: 3637 .....67
20:58:37.594477 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
0x0000: 0000 0000 0001 0000 0000 0002 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0002 0a00 0002 .....
0x0020: 0000 0000 0000 0a00 0001 .....
20:58:37.595793 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
0x0000: 0000 0000 0002 0000 0000 0001 0806 0001 .....
0x0010: 0800 0604 0001 0000 0000 0001 0a00 0001 .....
0x0020: 0000 0000 0000 0a00 0002 .....
20:58:37.595913 ARP, Reply 10.0.0.2 is-at 00:00:00:00:00:02, length 28
0x0000: 0000 0000 0001 0000 0000 0002 0806 0001 .....
0x0010: 0800 0604 0002 0000 0000 0002 0a00 0002 .....
0x0020: 0000 0000 0001 0a00 0001 .....
20:58:37.596030 ARP, Reply 10.0.0.1 is-at 00:00:00:00:00:01, length 28
0x0000: 0000 0000 0002 0000 0000 0001 0806 0001 .....
0x0010: 0800 0604 0002 0000 0000 0001 0a00 0001 .....
0x0020: 0000 0000 0002 0a00 0002 .....

```

H2 receiving the packets

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

```

root@ubuntu:/home/parth642001# tcpdump -D
1.h1-eth0 [Up, Running]
2.lo [Up, Running, Loopback]
3.any (Pseudo-device that captures on all interfaces) [Up, Running]
4.bluetooth-monitor (Bluetooth Linux Monitor) [none]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]

```

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

```

root@ubuntu:/home/parth642001# tcpdump -n -i h1-eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on h1-eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
21:06:01.646535 IP6 fe80::64df:1bff:fe21:4e96,5353 > ff02::fb,5353: 0 [2q] PTR (QM)? _ippe._tcp.local, PTR (QM)? _ipp._tcp.local. (45)
^C
1 packet captured
1 packet received by filter
0 packets dropped by kernel

```

- 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