Assignment 5

Implementation of L2 Forwarding and L3 Static Routing Protocol

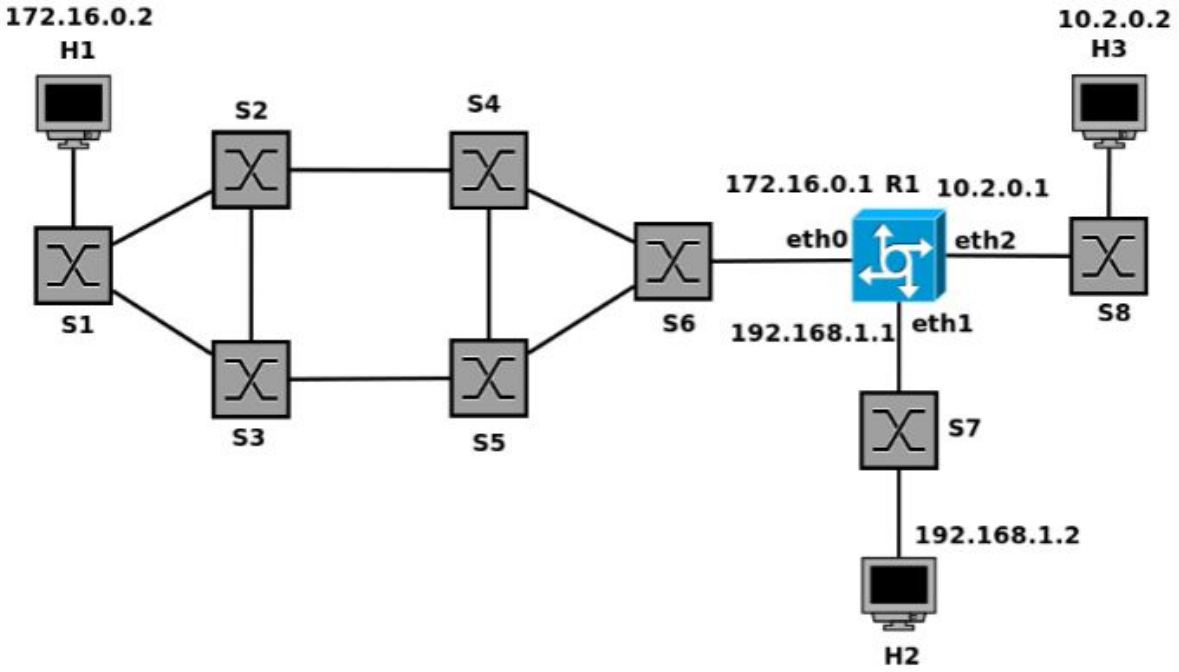
R.Sri Charan Reddy 14CS10037
G.Sai Bharath Chandra 14CS10020

Computer Networks Laboratory

Objective:

The objective of this assignment is to understand L2 Forwarding and L3 Routing mechanism using custom topologies in Mininet. The corresponding L2 forwarding and L3 routing protocols will be implemented as a part of the POX controller in Mininet.

Topology:



Instructions:

1. Copy topology ‘mytopo.py’ in mininet home folder.
2. Copy ‘l3_ass.py’ in pox/pox/forwarding folder.
3. Open two terminals.
4. Run ‘ssh -X mininet@localhost -p 8888’ mininet from both terminals
5. In terminal 1: Run ‘sudo mn --custom mytopo.py --topo mytopo --controller remote --switch ovsk --mac’

6. In terminal 2: Run ‘pox/pox.py --verbose openflow.spanning_tree --no-flood --hold-down openflow.discovery forwarding.l3_ass’
7. In terminal 1: Run ‘pingall’
8. Make sure that 6/6 are received and no packets dropped

Theory:

H1 constructs the IP packet and sets following fields in the IP header:
Source IP: 172.16.0.2
Destination IP: 10.2.0.2

H1 uses default gateway as its L3 next hop. So its next hop would be 172.16.0.1

It looks up the MAC table for the MAC address of 172.16.0.1. If the MAC address is there, it uses that MAC address. Otherwise it uses the ARP protocol to get the MAC address of 172.16.0.1

H1 broadcasts a ARP request corresponding to 172.16.0.1

Once R1-eth0 receives that ARP request, it sends back a ARP reply with its own MAC address

H1 then constructs the MAC header as follows:
Source MAC: MAC of H1
Destination MAC: MAC of R1-eth0 (corresponding to 172.16.0.1)

The packet is then forwarded to L2. It then uses the Spanning Tree Protocol(STP) to broadcast the packet at L2.

At L2, following the STP, every switch S1, S2, S3, S4, S5, S6 as well as R1-eth0 receives the packet. At every hop, the L2 device extracts the destination MAC address from the MAC header, and matches it with its own MAC address. Only R1- eth0 gets a match, and so it accepts the packet. R1-eth0 forwards the packet for IP processing at R1.

R1 looks up the destination IP in the packet header. Then it makes a destination look-up in the IP routing table. The routing table says that the next hop is 10.2.0.2 for the destination 10.2.0.2, and the interface for communication is R1-eth2. So this next hop and interface information is forwarded to the MAC layer of R1.

R1 again uses the MAC lookup (based on ARP) for 10.2.0.2. It constructs the MAC header as follows,
Source MAC: MAC for R1-eth2
Destination MAC: MAC of H3

The packet then again follows the L2 forwarding to eventually reach at H3.

Custom Topology:
The custom topology is created which specifies the default gateways of the host as well as declares the custom controllers. Moreover, the custom topo allocates the port on which the switches and routers should listen so as to register them to different controllers.

L3 forwarding:
It involves writing the controller protocol for the layer 3 routers.

L2 forwarding:

It involves writing the controller protocol for the layer 2 switches.

Observations and Discussions:

i) After step 5:

ii)After step 6:

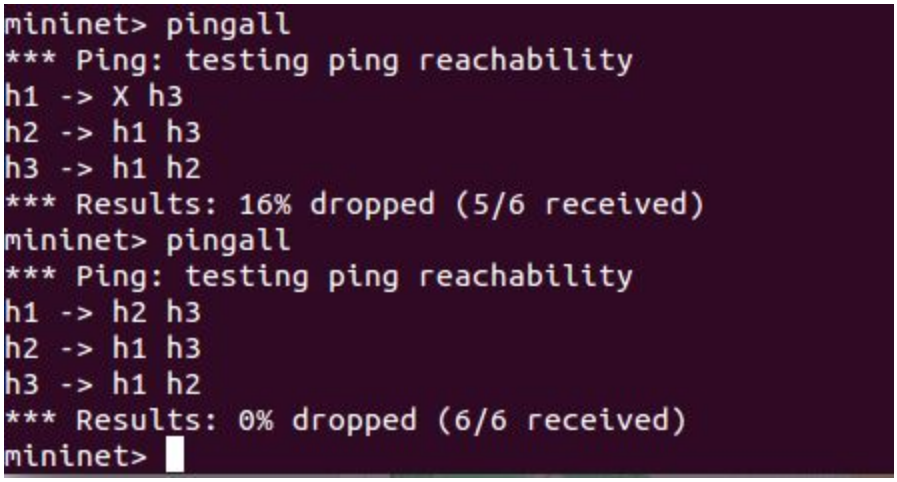
```
mininet@mininet-vm: ~
charan@gudda:~$ ssh  -X mininet@localhost -p 8888
mininet@localhost's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Tue Mar 21 00:23:02 2017 from 10.0.2.2
mininet@mininet-vm:~$ pox/pox.py  --verbose openflow.spanning_tree --no-flood --
hold-down openflow.discovery forwarding.l3_ass
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
DEBUG:openflow.spanning_tree:Spanning tree component ready
DEBUG:core:POX 0.2.0 (carp) going up...
DEBUG:core:Running on CPython (2.7.6/Mar 22 2014 22:59:56)
DEBUG:core:Platform is Linux-3.13.0-24-generic-x86_64-with-Ubuntu-14.04-trusty
INFO:core:POX 0.2.0 (carp) is up.
DEBUG:openflow.of_01:Listening on 0.0.0.0:6633
INFO:openflow.of_01:[00-00-00-00-00-05 1] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-05
DEBUG:openflow.spanning_tree:Disabling flooding for 4 ports
DEBUG:forwarding.l3_ass:-----------------------------------------------A Switch Co
nnectionUp!00-00-00-00-00-05-----------------------------------------------------
INFO:openflow.of_01:[00-00-00-00-00-08 4] connected
DEBUG:openflow.discovery:Installing flow for 00-00-00-00-00-08
DEBUG:openflow.spanning_tree:Disabling flooding for 3 ports
```

ii)<u>After step 7:</u>

```
mininet@mininet-vm: ~
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3
*** Adding switches:
r1 s2 s3 s4 s5 s6 s7 s8 s9
*** Adding links:
(h1, s2) (h2, s8) (r1, s9) (s2, s3) (s2, s4) (s3, s4) (s3, s5) (s
 (s5, s7) (s6, s7) (s7, r1) (s8, r1) (s9, h3)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 9 switches
r1 s2 s3 s4 s5 s6 s7 s8 s9 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

<u>Observations :-</u>

1). The first time when I ran pingall, some packets got dropped. But, when I did it next time, it was not the case. There was no loss of packets observed in subsequent pingall calls. The reason behind this is that for the first pingall, there is a delay to set up the paths and populating the tables as a result of which packets get dropped due to buffer overflow.



2). The TCP bandwidth decreases substantially when the hosts are not in the same subnet.Because, as we know, the switches are much faster than routers because of the route table lookup cost involved in L3 Processing while this is not the case for L2 Processing.