**CELL iteration 3 10/09/16: Task 3.4 Elective 3, POX NetApps, Dottie Kessler 10/2/16**

1. Start VM, login (mininet/mininet) and obtain ip (**sudo dhclient eth1**)
2. Run Xming
3. Putty to VM; run wireshark then filter of ; capture->interfaces->loopback (**sudo wireshare &**)
4. Putty to VM run ODL (**cd ~/opendaylight; sudo ./run.sh -virt ovsdb**)
5. Browser to host open IP of VM (admin/admin) [**http://192.168.56.101:8080**](http://192.168.56.101:8080)

**Task 3.4 Hands on Training with POX Python Net Apps elective 3 NORTHBOUND**

**Select three POX components (Stock)**

**POX**

1. POX functionality is provided by components.
2. You can specify multiple components on the command line
3. Not all components work well together, however, some depend on others
4. Some components take arguments, specified by 2 dashes
5. POX is the framework for communicating with SDN
6. POX can be used as a basic SDN controller
7. You can create more complex SDN controllers by creating your own new POX components
8. POX components are python programs
9. POX components implement network functionality in the SDN mininet network
10. Pox controller consists of 3 parts, Listener, control logic, messenger

**Task instructions**

1. Show how to setup the POX controller
2. Choose and run 3 stock python components
3. Test the operation of the control using mininet SDN
4. To run the POX Python scripts, go to the pox directory and use the following format to use the script:

**/home/mininet/freshpox/pox/pox.py directory.programName**

1. Start Mininet either before or after you run the POX Python programs

You can issue Mininet commands such as "pingall" and see the results in the POX program.

**sudo mn --topo=single,3 --controller=remote**

**Helpful**

<https://openflow.stanford.edu/display/ONL/POX+Wiki>

[http://www.brianlinkletter.com/using-pox-components-to-create-a-software-defined-networking-application**/**](http://www.brianlinkletter.com/using-pox-components-to-create-a-software-defined-networking-application/)

<https://haryachyy.wordpress.com/2014/06/14/learning-pox-openflow-controller-proactive-approach/>

<https://haryachyy.wordpress.com/page/2/>

<https://github.com/brandonheller/riplpox>

<http://kickstartsdn.com/ip_loadbalancer/>

<http://kickstartsdn.com>

### 1. COMPONENT: proactive\_net.py

1. Works proactively, depends on openflow discovery
2. Pushes paths when switches come up as opposed to reactive
3. Assign switches IP (dhcp), host must use assigned IP, forwarding based on IP
4. Relies on hosts sending DHCP requests to learn where the hosts are
5. You have to manually request dhcl
6. Knowledge of the IPS ahead of time, in advance, proactive, issued by DHCP
7. Routing routes get installed ahead of time
8. Redirect DHCP/LLDP/ARP/NDP packets to the controller
9. In this component, the routing is done with the forwarding.l2\_multi
10. **The Pox console shows the controller start and start mininet**

**sudo mn --topo=single,3 --controller=remote**

**./pox.py log.level openflow.of\_01 forwarding.topo\_proactive**  **openflow.discovery**

POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.

INFO:core:POX 0.2.0 (carp) is up.

INFO:openflow.of\_01:[00-00-00-00-00-01 1] connected

1. **dhclient**

From mininet

**mininet> h1 dhclient h1-eth0**

**mininet> h2 dhclient h2-eth0**

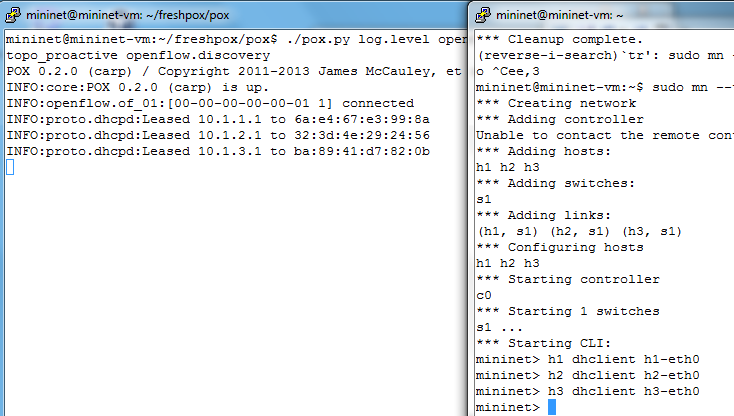
**mininet> h3 dhclient h3-eth0**

Response seen in the pox controller:

INFO:proto.dhcpd:Leased 10.1.1.1 to 6a:e4:67:e3:99:8a

INFO:proto.dhcpd:Leased 10.1.2.1 to 32:3d:4e:29:24:56

INFO:proto.dhcpd:Leased 10.1.3.1 to ba:89:41:d7:82:0b



1. **Dump flows (from putty window outside mininet/POX console)**

**sudo ovs-ofctl dump-flows s1**

NXST\_FLOW reply (xid=0x4):

cookie=0x0, duration=1380.713s, table=0, n\_packets=0, n\_bytes=0, idle\_age=1380, ip,nw\_dst=10.1.1.1 actions=mod\_dl\_src:00:00:00:00:00:01,mod\_dl\_dst:6a:e4:67:e3:99:8a,output:1

cookie=0x0, duration=1156.13s, table=0, n\_packets=0, n\_bytes=0, idle\_age=1156, ip,nw\_dst=10.1.3.1 actions=mod\_dl\_src:00:00:00:00:00:01,mod\_dl\_dst:ba:89:41:d7:82:0b,output:3

cookie=0x0, duration=1516.975s, table=0, n\_packets=0, n\_bytes=0, idle\_age=1516, priority=32767,ip,nw\_dst=255.255.255.255 actions=output:3,output:1,output:2

cookie=0x0, duration=1171.505s, table=0, n\_packets=0, n\_bytes=0, idle\_age=1171, ip,nw\_dst=10.1.2.1 actions=mod\_dl\_src:00:00:00:00:00:01,mod\_dl\_dst:32:3d:4e:29:24:56,output:2

cookie=0x0, duration=1516.975s, table=0, n\_packets=0, n\_bytes=0, idle\_age=1516, priority=32767,ip,nw\_dst=10.1.3.0/24 actions=CONTROLLER:65535

cookie=0x0, duration=1516.975s, table=0, n\_packets=0, n\_bytes=0, idle\_age=1516, priority=32767,ip,nw\_dst=10.1.1.0/24 actions=CONTROLLER:65535

cookie=0x0, duration=1516.975s, table=0, n\_packets=0, n\_bytes=0, idle\_age=1516, priority=32767,ip,nw\_dst=10.1.2.0/24 actions=CONTROLLER:65535

cookie=0x0, duration=1516.975s, table=0, n\_packets=0, n\_bytes=0, idle\_age=1516, priority=65000,dl\_dst=01:23:20:00:00:01,dl\_type=0x88cc actions=CONTROLLER:65535

cookie=0x0, duration=1516.975s, table=0, n\_packets=6, n\_bytes=2052, idle\_age=1156, udp,tp\_src=68,tp\_dst=67 actions=CONTROLLER:65535

cookie=0x0, duration=1516.975s, table=0, n\_packets=24, n\_bytes=1008, idle\_age=68, priority=28672,arp actions=CONTROLLER:65535

1. **Again using tree topo**

**sudo mn --controller=remote --topo tree,depth=2,fanout=8**

mininet> **net**

h1 h1-eth0:s2-eth1

h2 h2-eth0:s2-eth2

h3 h3-eth0:s2-eth3

h4 h4-eth0:s2-eth4

h5 h5-eth0:s2-eth5

h6 h6-eth0:s2-eth6

h7 h7-eth0:s2-eth7

h8 h8-eth0:s2-eth8

h9 h9-eth0:s3-eth1

h10 h10-eth0:s3-eth2

h11 h11-eth0:s3-eth3

h12 h12-eth0:s3-eth4

h13 h13-eth0:s3-eth5

h14 h14-eth0:s3-eth6

h15 h15-eth0:s3-eth7

h16 h16-eth0:s3-eth8

h17 h17-eth0:s4-eth1

h18 h18-eth0:s4-eth2

h19 h19-eth0:s4-eth3

h20 h20-eth0:s4-eth4

h21 h21-eth0:s4-eth5

h22 h22-eth0:s4-eth6

h23 h23-eth0:s4-eth7

h24 h24-eth0:s4-eth8

h25 h25-eth0:s5-eth1

h26 h26-eth0:s5-eth2

h27 h27-eth0:s5-eth3

h28 h28-eth0:s5-eth4

h29 h29-eth0:s5-eth5

h30 h30-eth0:s5-eth6

h31 h31-eth0:s5-eth7

h32 h32-eth0:s5-eth8

h33 h33-eth0:s6-eth1

h34 h34-eth0:s6-eth2

h35 h35-eth0:s6-eth3

h36 h36-eth0:s6-eth4

h37 h37-eth0:s6-eth5

h38 h38-eth0:s6-eth6

h39 h39-eth0:s6-eth7

h40 h40-eth0:s6-eth8

h41 h41-eth0:s7-eth1

h42 h42-eth0:s7-eth2

h43 h43-eth0:s7-eth3

h44 h44-eth0:s7-eth4

h45 h45-eth0:s7-eth5

h46 h46-eth0:s7-eth6

h47 h47-eth0:s7-eth7

h48 h48-eth0:s7-eth8

h49 h49-eth0:s8-eth1

h50 h50-eth0:s8-eth2

h51 h51-eth0:s8-eth3

h52 h52-eth0:s8-eth4

h53 h53-eth0:s8-eth5

h54 h54-eth0:s8-eth6

h55 h55-eth0:s8-eth7

h56 h56-eth0:s8-eth8

h57 h57-eth0:s9-eth1

h58 h58-eth0:s9-eth2

h59 h59-eth0:s9-eth3

h60 h60-eth0:s9-eth4

h61 h61-eth0:s9-eth5

h62 h62-eth0:s9-eth6

h63 h63-eth0:s9-eth7

h64 h64-eth0:s9-eth8

s1 lo: s1-eth1:s2-eth9 s1-eth2:s3-eth9 s1-eth3:s4-eth9 s1-eth4:s5-eth9 s1-eth5:s6-eth9 s1-eth6:s7-eth9 s1-eth7:s8-eth9 s1-eth8:s9-eth9

s2 lo: s2-eth1:h1-eth0 s2-eth2:h2-eth0 s2-eth3:h3-eth0 s2-eth4:h4-eth0 s2-eth5:h5-eth0 s2-eth6:h6-eth0 s2-eth7:h7-eth0 s2-eth8:h8-eth0 s2-eth9:s1-eth1

s3 lo: s3-eth1:h9-eth0 s3-eth2:h10-eth0 s3-eth3:h11-eth0 s3-eth4:h12-eth0 s3-eth5:h13-eth0 s3-eth6:h14-eth0 s3-eth7:h15-eth0 s3-eth8:h16-eth0 s3-eth9:s1-eth2

s4 lo: s4-eth1:h17-eth0 s4-eth2:h18-eth0 s4-eth3:h19-eth0 s4-eth4:h20-eth0 s4-eth5:h21-eth0 s4-eth6:h22-eth0 s4-eth7:h23-eth0 s4-eth8:h24-eth0 s4-eth9:s1-eth3

s5 lo: s5-eth1:h25-eth0 s5-eth2:h26-eth0 s5-eth3:h27-eth0 s5-eth4:h28-eth0 s5-eth5:h29-eth0 s5-eth6:h30-eth0 s5-eth7:h31-eth0 s5-eth8:h32-eth0 s5-eth9:s1-eth4

s6 lo: s6-eth1:h33-eth0 s6-eth2:h34-eth0 s6-eth3:h35-eth0 s6-eth4:h36-eth0 s6-eth5:h37-eth0 s6-eth6:h38-eth0 s6-eth7:h39-eth0 s6-eth8:h40-eth0 s6-eth9:s1-eth5

s7 lo: s7-eth1:h41-eth0 s7-eth2:h42-eth0 s7-eth3:h43-eth0 s7-eth4:h44-eth0 s7-eth5:h45-eth0 s7-eth6:h46-eth0 s7-eth7:h47-eth0 s7-eth8:h48-eth0 s7-eth9:s1-eth6

s8 lo: s8-eth1:h49-eth0 s8-eth2:h50-eth0 s8-eth3:h51-eth0 s8-eth4:h52-eth0 s8-eth5:h53-eth0 s8-eth6:h54-eth0 s8-eth7:h55-eth0 s8-eth8:h56-eth0 s8-eth9:s1-eth7

s9 lo: s9-eth1:h57-eth0 s9-eth2:h58-eth0 s9-eth3:h59-eth0 s9-eth4:h60-eth0 s9-eth5:h61-eth0 s9-eth6:h62-eth0 s9-eth7:h63-eth0 s9-eth8:h64-eth0 s9-eth9:s1-eth8

c0

**./pox.py log.level openflow.of\_01 forwarding.topo\_proactive openflow.discovery**

POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.

INFO:core:POX 0.2.0 (carp) is up.

INFO:openflow.of\_01:[00-00-00-00-00-01 1] connected

INFO:proto.dhcpd:Leased 10.1.1.1 to 6a:e4:67:e3:99:8a

INFO:proto.dhcpd:Leased 10.1.2.1 to 32:3d:4e:29:24:56

INFO:proto.dhcpd:Leased 10.1.3.1 to ba:89:41:d7:82:0b

WARNING:f.t\_p.00-00-00-00-00-01:6a:e4:67:e3:99:8a has incorrect IP 10.0.0.1

WARNING:f.t\_p.00-00-00-00-00-01:6a:e4:67:e3:99:8a has incorrect IP 10.0.0.1

WARNING:f.t\_p.00-00-00-00-00-01:6a:e4:67:e3:99:8a has incorrect IP 10.0.0.1

WARNING:f.t\_p.00-00-00-00-00-01:6a:e4:67:e3:99:8a has incorrect IP 10.0.0.1

WARNING:f.t\_p.00-00-00-00-00-01:6a:e4:67:e3:99:8a has incorrect IP 10.0.0.1

WARNING:f.t\_p.00-00-00-00-00-01:6a:e4:67:e3:99:8a has incorrect IP 10.0.0.1

WARNING:f.t\_p.00-00-00-00-00-01:6a:e4:67:e3:99:8a has incorrect IP 10.0.0.1

WARNING:f.t\_p.00-00-00-00-00-01:6a:e4:67:e3:99:8a has incorrect IP 10.0.0.1

WARNING:f.t\_p.00-00-00-00-00-01:6a:e4:67:e3:99:8a has incorrect IP 10.0.0.1

WARNING:f.t\_p.00-00-00-00-00-01:6a:e4:67:e3:99:8a has incorrect IP 10.0.0.1

<output truncated>

mininet> **h1 ping -c2 h2**

PING 10.2.2.1 (10.2.2.1) 56(84) bytes of data.

64 bytes from 10.2.2.1: icmp\_seq=1 ttl=64 time=0.209 ms

64 bytes from 10.2.2.1: icmp\_seq=2 ttl=64 time=0.100 ms

--- 10.2.2.1 ping statistics ---

2 packets transmitted, 2 received, 0% packet loss, time 999ms

rtt min/avg/max/mdev = 0.100/0.154/0.209/0.055 ms

### 2. COMPONENT: forwarding.l2\_pairs

1. Like l2\_learning, this component also makes OpenFlow switches act like a type of L2 learning
2. switch. Unlike l2\_learning, l2\_pairs installs rules based purely on MAC addresses.

* **./pox.py forwarding.l2\_pairs info.packet\_dump samples.pretty\_log log.level –DEBUG**
* **sudo mn --topo=single,3 --controller=remote**

1. **The Pox console shows the controller start**

mininet@mininet-vm:~/freshpox/pox$ **./pox.py forwarding.l2\_pairs info.packet\_dump samples.pretty\_log log.level --DEBUG**

POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.

INFO:forwarding.l2\_pairs:Pair-Learning switch running.

INFO:info.packet\_dump:Packet dumper running

[core ] POX 0.2.0 (carp) going up...

[core ] Running on CPython (2.7.6/Mar 22 2014 22:59:56)

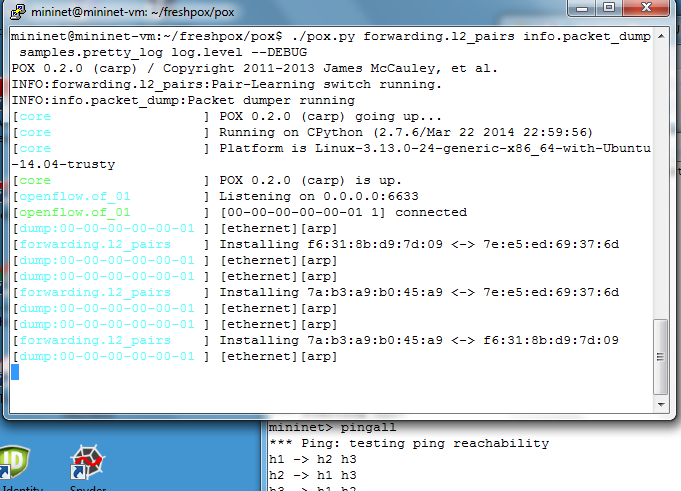
[core ] Platform is Linux-3.13.0-24-generic-x86\_64-with-Ubuntu-14.04-trusty

[core ] POX 0.2.0 (carp) is up.

[openflow.of\_01 ] Listening on 0.0.0.0:6633

[openflow.of\_01 ] [00-00-00-00-00-01 1] connected

1. **Test the controller, generate traffic** and see how it builds flows in the network. This generates traffic to the controller every time a switch receives a packet with a MAC destination that it does not already have in the flow table. When the controller (POX) receives a packet, it tells the switch to flood ARP to the other switches. The hosts will respond to the ARP request and then the POX sends openflow messages to the switches to load the flows



1. **Check the flow tables on all the switches**

mininet> **dpctl dump-flows**

\*\*\* s1 ------------------------------------------------------------------------

NXST\_FLOW reply (xid=0x4):

cookie=0x0, duration=710.399s, table=0, n\_packets=4, n\_bytes=280, idle\_age=705, dl\_src=7a:b3:a9:b0:45:a9,dl\_dst=7e:e5:ed:69:37:6d actions=output:1

cookie=0x0, duration=710.37s, table=0, n\_packets=3, n\_bytes=238, idle\_age=705, dl\_src=f6:31:8b:d9:7d:09,dl\_dst=7a:b3:a9:b0:45:a9 actions=output:3

cookie=0x0, duration=710.481s, table=0, n\_packets=3, n\_bytes=238, idle\_age=705, dl\_src=7e:e5:ed:69:37:6d,dl\_dst=f6:31:8b:d9:7d:09 actions=output:2

cookie=0x0, duration=710.443s, table=0, n\_packets=4, n\_bytes=280, idle\_age=705, dl\_src=f6:31:8b:d9:7d:09,dl\_dst=7e:e5:ed:69:37:6d actions=output:1

cookie=0x0, duration=710.331s, table=0, n\_packets=4, n\_bytes=280, idle\_age=705, dl\_src=7a:b3:a9:b0:45:a9,dl\_dst=f6:31:8b:d9:7d:09 actions=output:2

cookie=0x0, duration=710.438s, table=0, n\_packets=3, n\_bytes=238, idle\_age=705, dl\_src=7e:e5:ed:69:37:6d,dl\_dst=7a:b3:a9:b0:45:a9 actions=output:3

1. **Check the ARP tables of the hosts**

mininet> **h1 arp**

Address HWtype HWaddre Flags Mask Iface

10.0.0.3 ether 7a:b3:a9:b0:45:a9 C h1-eth0

10.0.0.2 ether f6:31:8b:d9:7d:09 C h1-eth0

mininet> **h2 arp**

Address HWtype HWaddress Flags Mask Iface

10.0.0.3 ether 7a:b3:a9:b0:45:a9 C h2-eth0

10.0.0.1 ether 7e:e5:ed:69:37:6d C h2-eth0

mininet> **h3 arp**

Address HWtype HWaddress Flags Mask Iface

10.0.0.2 ether f6:31:8b:d9:7d:09 C h3-eth0

10.0.0.1 ether 7e:e5:ed:69:37:6d C h3-eth0

1. **Clear the flows**

mininet> **dpctl del-flows**

\*\*\* s1 --------------------------------------------------------

mininet>

### 3. COMPONENT: misc.ip\_loadbalancer

1. simple TCP load balancer
2. a service\_ip and a list of addresses, the tcp flows to the service IP will round robin
3. servers will be probed periodically to make sure they are still alive, ARPs
4. The load balancer will be the first switch connected to
5. It does the l2\_learning, but ignores a particular switch, using the first switch as load balancer

* **pox.py misc.ip\_loadbalancer--ip=<Service IP>**

**--servers=<Server1 IP>,<Server2 IP>,… [–dpid=<dpid>]**

* **sudo mn --topo single,6 --mac --arp --controller=remote**

1. **Start mininet, 6 hosts, 1 switch, 1 controller**

**mininet parameters**

--mac makes the mac address of mininet hosts the same as their node number

--arp installs static ARP entries in all hosts

--switch ovsk uses Open vSwitch in kernel mode for each of the switches

--controller remote the SDN controller will run outside of mininet

--ip <host\_ip> switches should connect to the controller running on your local machine

mininet@mininet-vm:~$ **sudo mn --topo single,6 --mac --arp --controller=remote**

\*\*\* Creating network

\*\*\* Adding controller

Unable to contact the remote controller at 127.0.0.1:6633

\*\*\* Adding hosts:

h1 h2 h3 h4 h5 h6

\*\*\* Adding switches:

s1

\*\*\* Adding links:

(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1)

\*\*\* Configuring hosts

h1 h2 h3 h4 h5 h6

\*\*\* Starting controller

c0

\*\*\* Starting 1 switches

s1 ...

\*\*\* Starting CLI:

1. **Start Pox, use 10.0.1.1 for service id, 10.0.0.1, 10.0.0.2 for the http servers**

mininet@mininet-vm:~/freshpox/pox$ **./pox.py log.level --DEBUG misc.ip\_loadbalancer --ip=10.0.1.1 --servers=10.0.0.1,10.0.0.2**

POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.

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:[00-00-00-00-00-01 1] connected

INFO:iplb:IP Load Balancer Ready.

INFO:iplb:Load Balancing on [00-00-00-00-00-01 1]

INFO:iplb.00-00-00-00-00-01:Server 10.0.0.1 up

INFO:iplb.00-00-00-00-00-01:Server 10.0.0.2 up

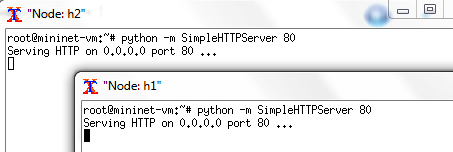
1. **Bring up some xterm windows, don’t forget this runs from within mininet (run XMING first)**

mininet> **xterm h1 h2 h3 h4 h5 h6**

mininet>

1. **Run a simple webserver on h1 and h2, port 8080 http**

**python -m SimpleHTTPServer 80**



1. **Now the other hosts can send requests to 10.0.1.1, see the traffic pattern on servers and controller**

On the xterm windows, for h3,h4,h5,h6: **curl 10.0.1.1**

