Software Defined Network (SDN) experiment using Mininet and POX Controller

Chih-Heng Ke (柯志亨)
Associate Professor, CSIE, National Quemoy University, Kimen, Taiwan
smallko@gmail.com

Outline

- Lab1: basic mininet operations
- Lab2: manually control the switch
- Lab3: move the rules to the POX controller
- Lab4: set different forwarding rules for each switch in the controller

Lab 1: basic mininet operations lab1.py

"""Custom topology example Two directly connected switches plus a host for each switch:
host --- switch --- switch --- host
Adding the 'topos' dict with a key/value pair to generate our newly defined topology enables one to pass in '--topo=mytopo' from the command line. """“

```
from mininet.topo import Topo

class MyTopo( Topo ):
  "Simple topology example."
  def __init__( self ):
    "Create custom topo." # Initialize topology
    Topo.__init__( self )
    # Add hosts and switches
    leftHost = self.addHost( 'h1' )
    rightHost = self.addHost( 'h2' )
    leftSwitch = self.addSwitch( 's3' )
    rightSwitch = self.addSwitch( 's4' )
    # Add links
    self.addLink( leftHost, leftSwitch )
    self.addLink( leftSwitch, rightSwitch )
    self.addLink( rightSwitch, rightHost )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

```
mininet@mininet-vm: ~/mylab
File  Edit  View  Search  Terminal  Help
mininet@mininet-vm:~/mylab$ pwd
/home/mininet/mylab
mininet@mininet-vm:~/mylab$ sudo mn --custom lab1.py --topo mytopo
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s3 s4
*** Adding links:
(h1, s3) (h2, s4) (s3, s4)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 2 switches
s3 s4
*** Starting CLI:
mininet>
```

The OpenFlow reference controller is used.

Display Mininet CLI commands:

```
mininet> help

Documented commands (type help <topic>):
========================================
EOF     exit    intfs     link    noecho       pingpair      py    source  xterm
dpctl   gterm   iperf     net     pingall      pingpairfull  quit  time
dump    help    iperfudp  nodes   pingallfull  px            sh    x

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2

mininet>
```

Display nodes:

```
mininet> nodes
available nodes are:
c0 h1 h2 s3 s4
mininet>
```

Display links:

```
mininet> net
h1 h1-eth0:s3-eth1
h2 h2-eth0:s4-eth2
s3 lo:  s3-eth1:h1-eth0 s3-eth2:s4-eth1
s4 lo:  s4-eth1:s3-eth2 s4-eth2:h2-eth0
c0
mininet>
```

Dump information about all nodes:

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=3874>
<Host h2: h2-eth0:10.0.0.2 pid=3875>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=3878>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None pid=3883>
<OVSController c0: 127.0.0.1:6633 pid=3866>
mininet>
```

Run a command on a host process:

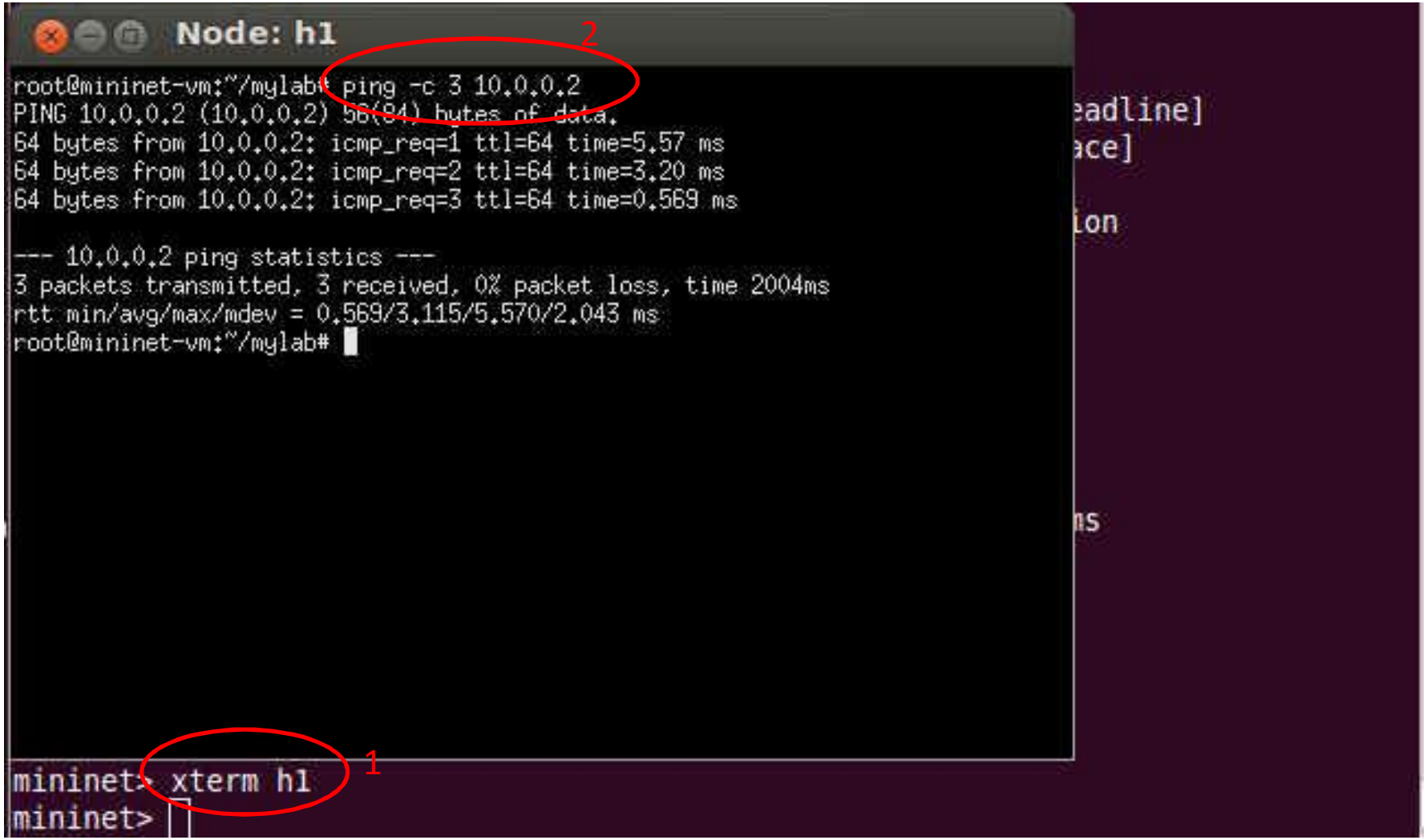
```
mininet> h1 ifconfig -a
h1-eth0   Link encap:Ethernet  HWaddr fa:a0:66:2d:4e:43
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::f8a0:66ff:fe2d:4e43/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:62 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10290 (10.2 KB)  TX bytes:846 (846.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet>
```

Tests connectivity between hosts

```
mininet> h1 ping -c 3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_req=1 ttl=64 time=7.20 ms
64 bytes from 10.0.0.2: icmp_req=2 ttl=64 time=1.00 ms
64 bytes from 10.0.0.2: icmp_req=3 ttl=64 time=0.138 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 0.138/2.782/7.207/3.148 ms
mininet>
```

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

Open an xterm for host h1 and test connectivity between h1 and h2.



Measure the bandwidth between hosts using iperf

mininet@mininet-vm: ~/mylab

File Edit View Search Terminal Help

```
mininet> h1 ping -c -3 h2
```

Node: h2

2

```
root@mininet-vm:~/mylab# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.2 port 5001 connected with 10.0.0.1 port 39810
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  1.12 GBytes   961 Mbits/sec
```

Node: h1

3

```
root@mininet-vm:~/mylab# iperf -c 10.0.0.2
------------------------------------------------------------
Client connecting to 10.0.0.2, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 10.0.0.1 port 39810 connected with 10.0.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec  1.12 GBytes   961 Mbits/sec
root@mininet-vm:~/mylab#
```

1

```
mininet> xterm h1 h2
mininet>
```

Exit Mininet

```
mininet> exit
*** Stopping 3 terms
*** Stopping 2 switches
s3 ..s4 ..
*** Stopping 2 hosts
h1 h2
*** Stopping 1 controllers
c0
*** Done
completed in 2361.770 seconds
mininet@mininet-vm:~/mylab$
```

Lab 2: manually control the switch

DPCTL

- dpctl: It is a command-line utility that sends basic OpenFlow messages to a switch
  - View switch port and flow statistics
  - View flow entries (FlowMods)
  - Add and delete FlowMods
- Useful tool for learning and debugging
- dpctl communicates directly with the switch and does not need a controller

Dpctl command list

show SWITCH	show basic information
status SWITCH [KEY]	report statistics (about KEY) (Not on HP)
show-protostat SWITCH	report protocol statistics (Not on HP)
dump-desc SWITCH	print switch description
dump-tables SWITCH	print table stats
mod-port SWITCH IFACE ACT	modify port behavior
dump-ports SWITCH [PORT]	print port statistics
desc SWITCH STRING	set switch description
dump-flows SWITCH	print all flow entries
dump-flows SWITCH FLOW	print matching FLOWs
dump-aggregate SWITCH	print aggregate flow statistics
dump-aggregate SWITCH FLOW	print aggregate stats for FLOWs
add-flow SWITCH FLOW	add flow described by FLOW
add-flows SWITCH FILE	add flows from FILE
mod-flows SWITCH FLOW	modify actions of matching FLOWs
del-flows SWITCH [FLOW]	delete matching FLOWs
monitor SWITCH	print packets received from SWITCH
execute SWITCH CMD [ARG...]	execute CMD with ARGS on SWITCH

Dpctl example usage

Flow fields and syntax:

- nw_tos=tos/dscp
- tp_dst=port
- icmp_type=type
- icmp_code=code

The following shorthand notations are also available:

- ip Same as dl_type=0x0800
- icmp Same as dl_type=0x0800,nw_proto=1
- tcp Same as dl_type=0x0800,nw_proto=6
- udp Same as dl_type=0x0800,nw_proto=17
- arp Same as dl_type=0x0806

Dpctl example usage

Flow fields and syntax:

- in_port=port_no
- dl_vlan=vlanID
- dl_src=mac
- dl_dst=mac
- dl_type=ethertype
- nw_src=ip[/netmask]
- nw_dst=ip[/netmask]
- nw_proto=proto

$ dpctl dump-flows tcp:15.255.124.107:6633

- Gives us information about the flows installed
- Rule itself
- Timeouts
- Actions
- Packets and bytes processed by flow

$ dpctl dump-ports tcp:15.255.124.107:6633

- Gives physical port information
- Rx, Tx counters
- Error counters

$ dpctl **mod-port** tcp:15.255.124.107:6633 17 down

Allows manipulation of the switch ports

- Up
- Down
- Flood
- Noflood

$ dpctl mod-port tcp:15.255.124.107:6633 2 down

Ping should fail now

$ dpctl mod-port tcp:15.255.124.107:6633 2 up

Ping works again

Let us add some flow entries so we can ping from host1 to host2

Test to ping Host 1 from Host 2 (should fail as we do not have any flow entries yet)

Add the flow entries (change port numbers):

$ dpctl add-flow tcp:15.255.124.107:6633 in_port=10,actions=output:14

$ dpctl add-flow tcp:15.255.124.107:6633 in_port=14,actions=output:10

Ping should work now!

Let us add some flow entries so we can ping from host1 to host2

Test to ping Host 1 from Host 2 (should fail as we do not have any flow entries yet)

Add the flow entries (change port numbers):

$ dpctl add-flow tcp:15.255.124.107:6633 in_port=10,actions=output:14

$ dpctl add-flow tcp:15.255.124.107:6633 in_port=14,actions=output:10

Ping should work now!

Following flow entries should now be shown:

$ dpctl dump-flows tcp:15.255.124.107:6634

- stats_reply (xid=0xd7d42712): flags=none type=1(flow)
- cookie=0, duration_sec=21s, duration_nsec=0s, table_id=2, priority=32768, n_packets=0, n_bytes=0, idle_timeout=60, hard_timeout=0, arp, actions=NORMAL
- cookie=0, duration_sec=7s, duration_nsec=36000000s, table_id=0, priority=32768, n_packets=0, n_bytes=0, idle_timeout=60, hard_timeout=0,ip,nw_dst=10.10.10.1,actions=output:2
- cookie=0, duration_sec=3s, duration_nsec=954000000s, table_id=0, priority=32768, n_packets=0, n_bytes=0, idle_timeout=60, hard_timeout=0,ip,nw_dst=10.10.10.2,actions=output:17

$ dpctl dump-ports tcp:15.255.124.107:6633

- stats_reply (xid=0xb2eeb981): flags=none type=4(port)
- 3 ports
- port 2: rx pkts=2756, bytes=527428, drop=0, errs=0, frame=?, over=?, crc=? tx pkts=2721, bytes=523911, drop=0, errs=0, coll=?
- port 17: rx pkts=2733, bytes=525187, drop=0, errs=0, frame=?, over=?, crc=? tx pkts=2727, bytes=525296, drop=0, errs=0, coll=?
- port 65534: rx pkts=?, bytes=?, drop=?, errs=?, frame=?, over=?, crc=? tx pkts=?, bytes=?, drop=?, errs=?, coll=?

Lets change the priority of flow

```
$ dpctl add-flow tcp:15.255.124.107:6633
ip,nw_dst=10.10.10.1,priority=1,actions=output:2
$ dpctl add-flow tcp:15.255.124.107:6633
ip,nw_dst=10.10.10.2,priority=2,actions=output:17
```

Lets see the flows in the switch

```
$ dpctl dump-flows tcp:15.255.124.107:6634
```

```
stats_reply (xid=0x8422afe4): flags=none type=1(flow)
cookie=0, duration_sec=3s, duration_nsec=899000000s, table_id=0, priority=1,
n_packets=0, n_bytes=0,
idle_timeout=60,hard_timeout=0,ip,nw_dst=10.10.10.1,actions=output:2
cookie=0, duration_sec=16s, duration_nsec=882000000s, table_id=0, priority=2,
n_packets=0, n_bytes=0,
idle_timeout=60,hard_timeout=0,ip,nw_dst=10.10.10.2,actions=output:17
```

Lab 2-1

```
mininet@mininet-vm: ~/mylab
File  Edit  View  Search  Terminal  Help
mininet@mininet-vm:~/mylab$ pwd
/home/mininet/mylab
mininet@mininet-vm:~/mylab$ sudo mn --custom lab1.py --topo mytopo
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s3 s4
*** Adding links:
(h1, s3) (h2, s4) (s3, s4)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 2 switches
s3 s4
*** Starting CLI:
mininet>
```

Set the rules for s3 and s4

```
mininet> s3 dpctl add-flow tcp:127.0.0.1:6634 in_port=1,actions=output:2
mininet> s3 dpctl add-flow tcp:127.0.0.1:6634 in_port=2,actions=output:1
mininet> s4 dpctl add-flow tcp:127.0.0.1:6634 in_port=1,actions=output:2
mininet> s4 dpctl add-flow tcp:127.0.0.1:6634 in_port=2,actions=output:1
```

Record what h1 has sent or received

```
mininet> h1 tcpdump -U -w /tmp/mylog &
```

Test connectivity between h1 and h2

```
mininet> h1 ping -c 5 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_req=1 ttl=64 time=4.16 ms
64 bytes from 10.0.0.2: icmp_req=2 ttl=64 time=0.726 ms
64 bytes from 10.0.0.2: icmp_req=3 ttl=64 time=0.211 ms
64 bytes from 10.0.0.2: icmp_req=4 ttl=64 time=0.154 ms
64 bytes from 10.0.0.2: icmp_req=5 ttl=64 time=0.192 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 0.154/1.090/4.168/1.553 ms
```

dump-flows results from s3 and s4

```
mininet> s3 dpctl dump-flows tcp:127.0.0.1:6634
stats_reply (xid=0xe77fe648): flags=none type=1(flow)
  cookie=0, duration_sec=42s, duration_nsec=102000000s, table_id=0, priority=327
68, n_packets=7, n_bytes=574, idle_timeout=60,hard_timeout=0,in_port=1,actions=o
utput:2
  cookie=0, duration_sec=37s, duration_nsec=330000000s, table_id=0, priority=327
68, n_packets=8, n_bytes=675, idle_timeout=60,hard_timeout=0,in_port=2,actions=o
utput:1
```

```
mininet> s4 dpctl dump-flows tcp:127.0.0.1:6634
stats_reply (xid=0x59dc1869): flags=none type=1(flow)
  cookie=0, duration_sec=56s, duration_nsec=218000000s, table_id=0, priority=327
68, n_packets=7, n_bytes=574, idle_timeout=60,hard_timeout=0,in_port=1,actions=o
utput:2
  cookie=0, duration_sec=51s, duration_nsec=446000000s, table_id=0, priority=327
68, n_packets=8, n_bytes=675, idle_timeout=60,hard_timeout=0,in_port=2,actions=o
utput:1
```

DPID: Unique identifier assigned by the switch for this OpenFlow instance

Number of tables and buffer size

```
mininet> s3 dpctl show tcp:127.0.0.1:6634
features_reply (xid=0x3755d761): ver:0x1, dpid:3
n_tables:255, n_buffers:256
features: capabilities:0xc7, actions:0xfff
 1(s3-eth1): addr:8e:0d:b3:a3:4b:83, config: 0, state:0
     current:    10GB-FD COPPER
 2(s3-eth2): addr:0a:00:0a:14:6f:67, config: 0, state:0
     current:    10GB-FD COPPER
 LOCAL(s3): addr:12:1e:2c:b9:59:4f, config: 0x1, state:0x1
get config reply (xid=0x4cbfa971): miss send len=0
```

Port Information

Use wireshark to see what h1 has sent or received

```
mininet> exit
*** Stopping 2 switches
s3 ..s4 ..
*** Stopping 2 hosts
h1 h2
*** Stopping 1 controllers
c0
*** Done
completed in 122.543 seconds
mininet@mininet-vm:~/mylab$ wireshark /tmp/mylog
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	fe80::38eb:10ff:fe78:5a9	ff02::fb	MDNS	101	Standard query PTR _sar
2	0.056214	fe80::9cde:a0ff:fe52:459	ff02::fb	MDNS	101	Standard query PTR _san
3	31.992135	fe80::38eb:10ff:fe78:5a9	ff02::fb	MDNS	101	Standard query PTR _san
4	32.052820	fe80::9cde:a0ff:fe52:459	ff02::fb	MDNS	101	Standard query PTR _san
5	80.496689	be:a8:05:42:48:5d	Broadcast	ARP	42	Who has 10.0.0.2? Tell
6	80.500386	d2:68:ea:f9:93:fd	be:a8:05:42:48:5d	ARP	42	10.0.0.2 is at d2:68:ea
7	80.500548	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id
8	80.504051	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id
9	81.497144	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id
10	81.497927	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id
11	82.498822	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id
12	82.499066	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id
13	83.497789	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id
14	83.497874	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id
15	84.497068	10.0.0.1	10.0.0.2	ICMP	98	Echo (ping) request id
16	84.497185	10.0.0.2	10.0.0.1	ICMP	98	Echo (ping) reply id
17	85.519119	d2:68:ea:f9:93:fd	be:a8:05:42:48:5d	ARP	42	Who has 10.0.0.1? Tell
18	85.519158	be:a8:05:42:48:5d	d2:68:ea:f9:93:fd	ARP	42	10.0.0.1 is at be:a8:05

Lab 2-2

mininet@mininet-vm: ~/mylab

File Edit View Search Terminal Help

```
mininet@mininet-vm:~/mylab$ pwd
/home/mininet/mylab
mininet@mininet-vm:~/mylab$ sudo mn --custom lab1.py --topo mytopo
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s3 s4
*** Adding links:
(h1, s3) (h2, s4) (s3, s4)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 2 switches
s3 s4
*** Starting CLI:
mininet>
```

Set the rules for s3

```
mininet> s3 dpctl add-flow tcp:127.0.0.1:6634 idle_timeout=0,hard_timeout=0,prio
rity=1,in_port=1,actions=output:2
mininet> s3 dpctl add-flow tcp:127.0.0.1:6634 idle_timeout=0,hard_timeout=0,prio
rity=1,in_port=2,actions=output:1
mininet> s3 dpctl add-flow tcp:127.0.0.1:6634 idle_timeout=0,hard_timeout=0,prio
rity=10,ip,nw_dst=10.0.0.1,actions=output:1
mininet> s3 dpctl add-flow tcp:127.0.0.1:6634 idle_timeout=0,hard_timeout=0,prio
rity=10,ip,nw_dst=10.0.0.2,actions=output:2
mininet> s3 dpctl dump-flows tcp:127.0.0.1:6634
stats_reply (xid=0xdb9daac4): flags=none type=1(flow)
  cookie=0, duration_sec=53s, duration_nsec=754000000s, table_id=0, priority=1,
n_packets=0, n_bytes=0, idle_timeout=0,hard_timeout=0,in_port=1,actions=output:2
  cookie=0, duration_sec=48s, duration_nsec=460000000s, table_id=0, priority=1, n
_packets=0, n_bytes=0, idle_timeout=0,hard_timeout=0,in_port=2,actions=output:1
  cookie=0, duration_sec=11s, duration_nsec=711000000s, table_id=0, priority=10,
 n_packets=0, n_bytes=0, idle_timeout=0,hard_timeout=0,ip,nw_dst=10.0.0.1,action
s=output:1
  cookie=0, duration_sec=6s, duration_nsec=740000000s, table_id=0, priority=10,
n_packets=0, n_bytes=0, idle_timeout=0,hard_timeout=0,ip,nw_dst=10.0.0.2,actions
=output:2
```

Set the rules for s4

```
mininet> s4 dpctl add-flow tcp:127.0.0.1:6634 idle_timeout=0,hard_timeout=0,pri
rity=1,in_port=1,actions=output:2
mininet> s4 dpctl add-flow tcp:127.0.0.1:6634 idle_timeout=0,hard_timeout=0,pri
rity=1,in_port=2,actions=output:1
mininet> s4 dpctl add-flow tcp:127.0.0.1:6634 idle_timeout=0,hard_timeout=0,pri
rity=10,ip,nw_dst=10.0.0.1,actions=output:1
mininet> s4 dpctl add-flow tcp:127.0.0.1:6634 idle_timeout=0,hard_timeout=0,pri
rity=10,ip,nw_dst=10.0.0.2,actions=output:2
mininet> s4 dpctl dump-flows tcp:127.0.0.1:6634
stats_reply (xid=0x8f369983): flags=none type=1(flow)
  cookie=0, duration_sec=33s, duration_nsec=941000000s, table_id=0, priority=1,
n_packets=0, n_bytes=0, idle_timeout=0,hard_timeout=0,in_port=1,actions=output:2
  cookie=0, duration_sec=24s, duration_nsec=61000000s, table_id=0, priority=1, n
_packets=0, n_bytes=0, idle_timeout=0,hard_timeout=0,in_port=2,actions=output:1
  cookie=0, duration_sec=16s, duration_nsec=76000000s, table_id=0, priority=10,
n_packets=0, n_bytes=0, idle_timeout=0,hard_timeout=0,ip,nw_dst=10.0.0.1,actions
=output:1
  cookie=0, duration_sec=6s, duration_nsec=458000000s, table_id=0, priority=10,
n_packets=0, n_bytes=0, idle_timeout=0,hard_timeout=0,ip,nw_dst=10.0.0.2,actions
=output:2
```

Test connectivity between h1 and h2

```
mininet> h1 ping -c 5 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_req=1 ttl=64 time=3.88 ms
64 bytes from 10.0.0.2: icmp_req=2 ttl=64 time=0.327 ms
64 bytes from 10.0.0.2: icmp_req=3 ttl=64 time=0.157 ms
64 bytes from 10.0.0.2: icmp_req=4 ttl=64 time=0.187 ms
64 bytes from 10.0.0.2: icmp_req=5 ttl=64 time=0.350 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3998ms
rtt min/avg/max/mdev = 0.157/0.980/3.881/1.452 ms
```

Workspaces

arp

Ping (echo)

Ping (reply)

```
mininet> s3 dpctl dump-flows tcp:127.0.0.1:6634
stats_reply (xid=0x4c09a44a): flags=none type=1(flow)
  cookie=0, duration_sec=92s, duration_nsec=685000000s, table_id=0, priority=1,
n_packets=2, n_bytes=84, idle_timeout=0,hard_timeout=0,in_port=1,actions=output:
2
  cookie=0, duration_sec=82s, duration_nsec=805000000s, table_id=0, priority=1,
n_packets=3, n_bytes=185, idle_timeout=0,hard_timeout=0,in_port=2,actions=output
:1
  cookie=0, duration_sec=74s, duration_nsec=820000000s, table_id=0, priority=10,
 n_packets=5, n_bytes=490, idle_timeout=0,hard_timeout=0,ip,nw_dst=10.0.0.1,acti
ons=output:1
  cookie=0, duration_sec=65s, duration_nsec=202000000s, table_id=0, priority=10,
 n_packets=5, n_bytes=490, idle_timeout=0,hard_timeout=0,ip,nw_dst=10.0.0.2,acti
ons=output:2
```

```
mininet> s4 dpctl dump-flows tcp:127.0.0.1:6634
stats_reply (xid=0x94ca73e6): flags=none type=1(flow)
  cookie=0, duration_sec=145s, duration_nsec=3000000s, table_id=0, priority=1, n
_packets=2, n_bytes=84, idle_timeout=0,hard_timeout=0,in_port=1,actions=output:2
  cookie=0, duration_sec=135s, duration_nsec=123000000s, table_id=0, priority=1,
 n_packets=3, n_bytes=185, idle_timeout=0,hard_timeout=0,in_port=2,actions=outpu
t:1
  cookie=0, duration_sec=127s, duration_nsec=138000000s, table_id=0, priority=10
, n_packets=5, n_bytes=490, idle_timeout=0,hard_timeout=0,ip,nw_dst=10.0.0.1,act
ions=output:1
  cookie=0, duration_sec=117s, duration_nsec=520000000s, table_id=0, priority=10
, n_packets=5, n_bytes=490, idle_timeout=0,hard_timeout=0,ip,nw_dst=10.0.0.2,act
ions=output:2
```

Lab 3: move the rules to the POX controller

lab3_1.py

```
#!/usr/bin/python

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import CPULimitedHost
from mininet.link import TCLink
from mininet.util import dumpNodeConnections
from mininet.log import setLogLevel
from mininet.node import Controller

import os

class POXcontroller1( Controller):
  def start(self):
    self.pox='%s/pox/pox.py' %os.environ['HOME']
    self.cmd(self.pox, "lab3_1_controller &")
  def stop(self):
    self.cmd('kill %' +self.pox)

controllers = { 'poxcontroller1': POXcontroller1}
```

```
class SingleSwitchTopo(Topo):
  "Single switch connected to n hosts."
  def __init__(self, n=2, **opts):
    Topo.__init__(self, **opts)
    switch = self.addSwitch('s1')
    # Each host gets 50%/n of system CPU
    h1=self.addHost('h1', cpu=.5/n)
    h2=self.addHost('h2', cpu=.5/n)

    # 10 Mbps, 10ms delay, 0% loss, 1000
packet queue
    self.addLink('h1', switch, bw=10,
delay='10ms', loss=0,
max_queue_size=1000, use_htb=True)
    self.addLink('h2', switch, bw=10,
delay='10ms', loss=0,
max_queue_size=1000, use_htb=True)
```

```
def perfTest():
    "Create network and run simple performance test"
    topo = SingleSwitchTopo(n=2)
    net = Mininet(topo=topo, host=CPULimitedHost, link=TCLink, controller=POXcontroller1)
    net.start()
    print "Dumping host connections"
    dumpNodeConnections(net.hosts)
    print "Testing network connectivity"
    net.pingAll()
    print "Testing bandwidth between h1 and h2"
    h1, h2 = net.get('h1', 'h2')
    net.iperf((h1, h2))
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    perfTest()
```

lab3_1_controller.py

```
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.util import dpidToStr

log = core.getLogger()

def _handle_ConnectionUp (event):
  msg = of.ofp_flow_mod()
  msg.priority =1
  msg.idle_timeout = 0
  msg.hard_timeout = 0
  msg.match.in_port =1
  msg.actions.append(of.ofp_action_output(port = 2))
  event.connection.send(msg)

  msg = of.ofp_flow_mod()
  msg.priority =1
  msg.idle_timeout = 0
  msg.hard_timeout = 0
  msg.match.in_port =2
  msg.actions.append(of.ofp_action_output(port = 1))
  event.connection.send(msg)

def launch ():
  core.openflow.addListenerByName("ConnectionUp", _handle_ConnectionUp)
```

Put the lab3_1_controller.py under ~/pox/ext

```
mininet@mininet-vm:~/mylab$ pwd
/home/mininet/mylab
mininet@mininet-vm:~/mylab$ ls
lab1.py  lab3_1.py  lab3_1.py~  lab3_2.py  lab3_2.py~
mininet@mininet-vm:~/mylab$ sudo ./lab3_1.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit 10ms delay 0% loss) (10.00Mbit 10ms delay 0% loss) (h1, s1) (10.00Mbi
t 10ms delay 0% loss) (10.00Mbit 10ms delay 0% loss) (h2, s1)
*** Configuring hosts
h1 (cfs 25000/100000us) h2 (cfs 25000/100000us)
*** Starting controller
*** Starting 1 switches
s1 (10.00Mbit 10ms delay 0% loss) (10.00Mbit 10ms delay 0% loss)
Dumping host connections
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
Testing network connectivity
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
Testing bandwidth between h1 and h2
*** Iperf: testing TCP bandwidth between h1 and h2
waiting for iperf to start up...*** Results: ['9.11 Mbits/sec', '9.21 Mbits/sec'
]
*** Stopping 1 switches
s1 ..
*** Stopping 2 hosts
h1 h2
*** Stopping 1 controllers
c0
*** Done
```

lab3_2.py

```
from mininet.node import CPULimitedHost
from mininet.link import TCLink
from mininet.util import
dumpNodeConnections
from mininet.log import setLogLevel
from mininet.node import Controller

import os

class POXcontroller2( Controller):
  def start(self):
    self.pox='%s/pox/pox.py'
%os.environ['HOME']
    self.cmd(self.pox, "lab3_2_controller &")
  def stop(self):
    self.cmd('kill %' +self.pox)

controllers = { 'poxcontroller1': POXcontroller2}
```

```
class SingleSwitchTopo(Topo):
  "Single switch connected to n hosts."
  def __init__(self, n=2, **opts):
    Topo.__init__(self, **opts)
    switch = self.addSwitch('s1')
    # Each host gets 50%/n of system CPU
    h1=self.addHost('h1', cpu=.5/n)
    h2=self.addHost('h2', cpu=.5/n)

    # 10 Mbps, 10ms delay, 0% loss, 1000
packet queue
    self.addLink('h1', switch, bw=10,
delay='10ms', loss=0, max_queue_size=1000,
use_htb=True)
    self.addLink('h2', switch, bw=10,
delay='10ms', loss=0, max_queue_size=1000,
use_htb=True)
```

```
def perfTest():
    "Create network and run simple performance test"
    topo = SingleSwitchTopo(n=2)
    net = Mininet(topo=topo,
                  host=CPULimitedHost, link=TCLink,
controller=POXcontroller2)
    net.start()
    h1, h2 = net.get('h1', 'h2')
    h1.setIP( '192.168.123.1/24' )
    h2.setIP( '192.168.123.2/24' )
    print "Dumping host connections"
    dumpNodeConnections(net.hosts)
    print "Testing network connectivity"
    net.pingAll()
    print "Testing bandwidth between h1 and h2"
    #net.iperf((h1, h2))
    h2.cmd('iperf -s -u -i 1 > /tmp/lab3_2 &')
    print h1.cmd('iperf -c 192.168.123.2 -u -b 10m -t 10')
    h2.cmd('kill %iperf')
    f=open('/tmp/lab3_2')
    lineno=1
    for line in f.readlines():
      print "%d: %s" % (lineno, line.strip())
      lineno+=1
    net.stop()
```

```
if __name__ == '__main__':
    setLogLevel('info')
    perfTest()
```

lab3_2_controller.py

Put the lab3_2_controller.py under ~/pox/ext

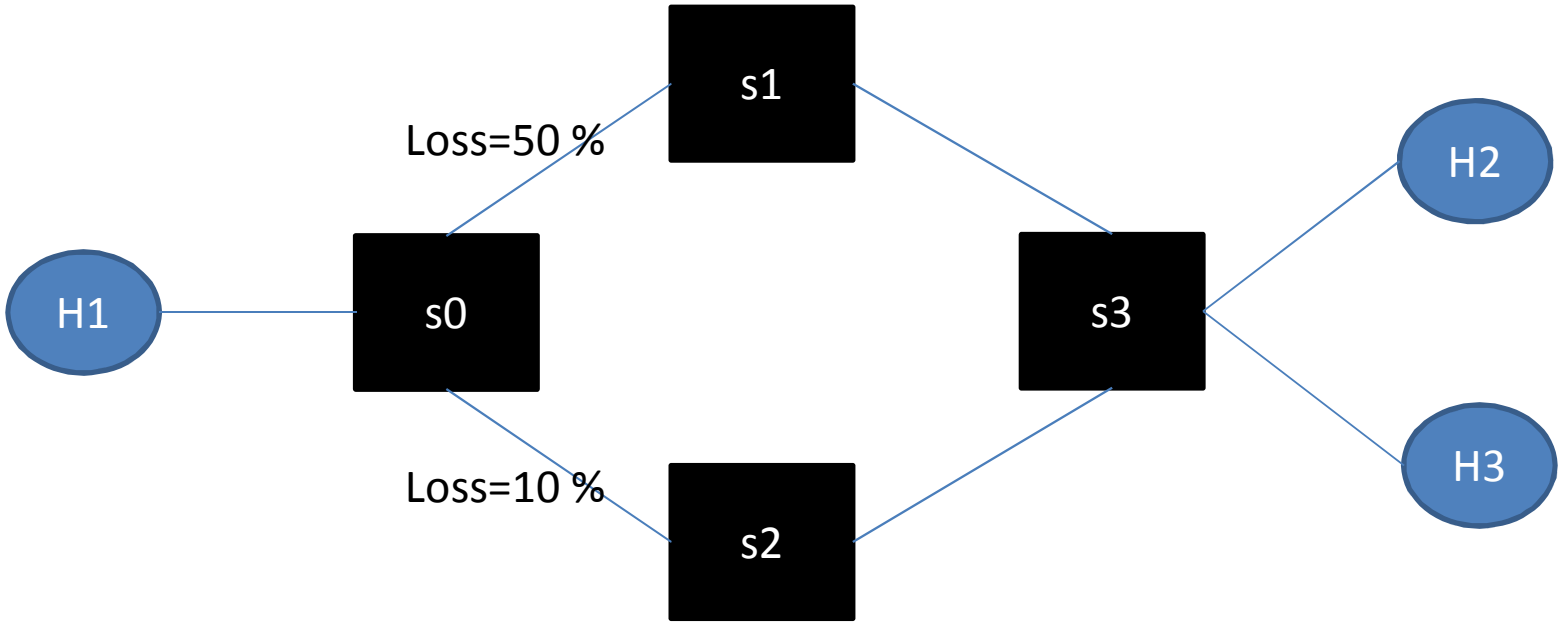
```
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.util import dpidToStr

log = core.getLogger()

def _handle_ConnectionUp (event):
  msg = of.ofp_flow_mod()
  msg.priority =1
  msg.idle_timeout = 0
  msg.hard_timeout = 0
  msg.match.in_port =1
  msg.actions.append(of.ofp_action_output(port = 2))
  event.connection.send(msg)

  msg = of.ofp_flow_mod()
  msg.priority =1
  msg.idle_timeout = 0
  msg.hard_timeout = 0
  msg.match.in_port =2
  msg.actions.append(of.ofp_action_output(port = 1))
  event.connection.send(msg)
```

```
  msg = of.ofp_flow_mod()
  msg.priority =10
  msg.idle_timeout = 0
  msg.hard_timeout = 0
  msg.match.dl_type = 0x0800
  msg.match.nw_dst = "192.168.123.2"
  msg.actions.append(of.ofp_action_output(port = 2))
  event.connection.send(msg)

  msg = of.ofp_flow_mod()
  msg.priority =10
  msg.idle_timeout = 0
  msg.hard_timeout = 0
  msg.match.dl_type = 0x0800
  msg.match.nw_dst = "192.168.123.1"
  msg.actions.append(of.ofp_action_output(port = 1))
  event.connection.send(msg)

def launch ():
  core.openflow.addListenerByName("ConnectionUp",
_handle_ConnectionUp)
```

```
mininet@mininet-vm:~/mylab$ sudo ./lab3_2.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit 10ms delay 0% loss) (10.00Mbit 10ms delay 0% loss) (h1, s1) (10.00Mbi
t 10ms delay 0% loss) (10.00Mbit 10ms delay 0% loss) (h2, s1)
*** Configuring hosts
h1 (cfs 25000/100000us) h2 (cfs 25000/100000us)
*** Starting controller
*** Starting 1 switches
s1 (10.00Mbit 10ms delay 0% loss) (10.00Mbit 10ms delay 0% loss)
Dumping host connections
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
Testing network connectivity
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
Testing bandwidth between h1 and h2
------------------------------------------------------------
Client connecting to 192.168.123.2, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
[  3] local 192.168.123.1 port 33574 connected with 192.168.123.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  11.9 MBytes  10.0 Mbits/sec
[  3] Sent 8503 datagrams
[  3] Server Report:
[  3]  0.0-10.6 sec  11.9 MBytes  9.45 Mbits/sec   1.402 ms    0/ 8502 (0%)
[  3]  0.0-10.6 sec  1 datagrams received out-of-order
```

```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  160 KByte (default)
------------------------------------------------------------
[  3] local 192.168.123.2 port 5001 connected with 192.168.123.1 port 33574
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagr
ams
[  3]  0.0- 1.0 sec  1.13 MBytes  9.50 Mbits/sec   0.591 ms    0/  808 (0%)
[  3]  1.0- 2.0 sec  1.15 MBytes  9.64 Mbits/sec   0.453 ms    0/  820 (0%)
[  3]  2.0- 3.0 sec  1.12 MBytes  9.43 Mbits/sec   0.306 ms    0/  802 (0%)
[  3]  3.0- 4.0 sec  1.14 MBytes  9.55 Mbits/sec   0.254 ms    0/  812 (0%)
[  3]  4.0- 5.0 sec  1.12 MBytes  9.37 Mbits/sec   0.518 ms    0/  797 (0%)
[  3]  5.0- 6.0 sec  1.11 MBytes  9.33 Mbits/sec   0.289 ms    0/  793 (0%)
[  3]  6.0- 7.0 sec  1.16 MBytes  9.73 Mbits/sec   0.295 ms    0/  827 (0%)
[  3]  7.0- 8.0 sec  1.16 MBytes  9.73 Mbits/sec   0.384 ms    0/  827 (0%)
[  3]  8.0- 9.0 sec  1.10 MBytes  9.22 Mbits/sec   0.306 ms    0/  784 (0%)
[  3]  9.0-10.0 sec  1.16 MBytes  9.73 Mbits/sec   0.425 ms    0/  827 (0%)
[  3]  0.0-10.6 sec  11.9 MBytes  9.45 Mbits/sec   1.403 ms    0/ 8502 (0%)
[  3]  0.0-10.6 sec  1 datagrams received out-of-order
*** Stopping 1 switches
s1 ..
*** Stopping 2 hosts
h1 h2
*** Stopping 1 controllers
c0
*** Done
```

Lab 4: set different forwarding rules for each switch in the controller



H1->H2: H1-s0-s1-s3-H2
H1->H3: H1-s0-s2-s3-H3

```
#!/usr/bin/python lab4.py

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import CPULimitedHost
from mininet.link import TCLink
from mininet.util import
dumpNodeConnections
from mininet.log import setLogLevel
from mininet.node import Controller
from mininet.cli import CLI

import os

class POXcontroller1( Controller):
  def start(self):
    self.pox='%s/pox/pox.py'
%os.environ['HOME']
    self.cmd(self.pox, "lab4_controller >
/tmp/lab4_controller &")
  def stop(self):
    self.cmd('kill %' +self.pox)

controllers = { 'poxcontroller': POXcontroller1}
```

```
class MyTopo(Topo):
  def __init__(self, n=2,**opts):
    Topo.__init__(self, **opts)
    s0 = self.addSwitch('s0')
    s1 = self.addSwitch('s1')
    s2 = self.addSwitch('s2')
    s3 = self.addSwitch('s3')
    h1=self.addHost('h1', cpu=.5/n)
    h2=self.addHost('h2', cpu=.5/n)
    h3=self.addHost('h3', cpu=.5/n)
    self.addLink(h1, s0, bw=10, delay='10ms',
loss=0, max_queue_size=1000, use_htb=True)
    self.addLink(s0, s1, bw=10, delay='10ms',
loss=50, max_queue_size=1000, use_htb=True)
    self.addLink(s0, s2, bw=10, delay='10ms',
loss=10, max_queue_size=1000, use_htb=True)
    self.addLink(s1, s3, bw=10, delay='10ms',
loss=0, max_queue_size=1000, use_htb=True)
    self.addLink(s2, s3, bw=10, delay='10ms',
loss=0, max_queue_size=1000, use_htb=True)
    self.addLink(s3, h2, bw=10, delay='10ms',
loss=0, max_queue_size=1000, use_htb=True)
    self.addLink(s3, h3, bw=10, delay='10ms',
loss=0, max_queue_size=1000, use_htb=True)
```

```
def perfTest():
    "Create network and run simple
performance test"
    topo = MyTopo(n=3)
    net = Mininet(topo=topo,
host=CPULimitedHost, link=TCLink,
controller=POXcontroller1)
    net.start()
    print "Dumping host connections"
    dumpNodeConnections(net.hosts)
    CLI(net)
    net.stop()

if __name__ == '__main__':
    setLogLevel('info')
    perfTest()
```

Command line interface

lab4_controller.py

Put the lab4_controller.py under ~/pox/ext

```
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.util import dpidToStr
log = core.getLogger()
s0_dpid=0
s1_dpid=0
s2_dpid=0
s3_dpid=0

def _handle_ConnectionUp (event):
  global s0_dpid, s1_dpid, s2_dpid, s3_dpid
  print "ConnectionUp: ", dpidToStr(event.connection.dpid)

 #remember the connection dpid for switch
  for m in event.connection.features.ports:
    if m.name == "s0-eth1":
      s0_dpid = event.connection.dpid
      print "s0_dpid=", s0_dpid
    elif m.name == "s1-eth1":
      s1_dpid = event.connection.dpid
      print "s1_dpid=", s1_dpid
    elif m.name == "s2-eth1":
      s2_dpid = event.connection.dpid
      print "s2_dpid=", s2_dpid
    elif m.name == "s3-eth1":
      s3_dpid = event.connection.dpid
      print "s3_dpid=", s3_dpid
```

```
def _handle_PacketIn (event):
  global s0_dpid, s1_dpid, s2_dpid, s3_dpid
  print "PacketIn: ", dpidToStr(event.connection.dpid)

  if event.connection.dpid==s0_dpid:
    msg = of.ofp_flow_mod()
    msg.priority =1
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.dl_type = 0x0806
    msg.actions.append(of.ofp_action_output(port = of.OFPP_ALL))
    event.connection.send(msg)

    msg = of.ofp_flow_mod()
    msg.priority =10
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.dl_type = 0x0800
    msg.match.nw_dst = "10.0.0.1"
    msg.actions.append(of.ofp_action_output(port = 1))
    event.connection.send(msg)
```

```
msg = of.ofp_flow_mod()
msg.priority =10
msg.idle_timeout = 0
msg.hard_timeout = 0
msg.match.dl_type = 0x0800
msg.match.nw_dst = "10.0.0.2"
msg.actions.append(of.ofp_action_output(port = 2))
event.connection.send(msg)

msg = of.ofp_flow_mod()
msg.priority =10
msg.idle_timeout = 0
msg.hard_timeout = 0
msg.match.dl_type = 0x0800
msg.match.nw_dst = "10.0.0.3"
msg.actions.append(of.ofp_action_output(port = 3))
event.connection.send(msg)
```

```
elif event.connection.dpid==s1_dpid:
    msg = of.ofp_flow_mod()
    msg.priority =1
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.in_port =1
msg.actions.append(of.ofp_action_output(port =
2))
    event.connection.send(msg)

    msg = of.ofp_flow_mod()
    msg.priority =1
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.in_port =2

msg.actions.append(of.ofp_action_output(port =
1))
    event.connection.send(msg)
```

```
elif event.connection.dpid==s2_dpid:
    msg = of.ofp_flow_mod()
    msg.priority =1
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.in_port =1
msg.actions.append(of.ofp_action_outp
ut(port = 2))
    event.connection.send(msg)

    msg = of.ofp_flow_mod()
    msg.priority =1
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.in_port =2
msg.actions.append(of.ofp_action_outp
ut(port = 1))
    event.connection.send(msg)
```

```
elif event.connection.dpid==s3_dpid:
    msg = of.ofp_flow_mod()
    msg.priority =1
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.dl_type = 0x0806
    msg.actions.append(of.ofp_action_output(port = of.OFPP_ALL))
    event.connection.send(msg)

    msg = of.ofp_flow_mod()
    msg.priority =10
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.dl_type = 0x0800
    msg.match.nw_dst = "10.0.0.2"
    msg.actions.append(of.ofp_action_output(port = 3))
    event.connection.send(msg)
                msg = of.ofp_flow_mod()
                msg.priority =10
                msg.idle_timeout = 0
                msg.hard_timeout = 0
                msg.match.dl_type = 0x0800
                msg.match.nw_dst = "10.0.0.3"
                msg.actions.append(of.ofp_action_output(port = 4))
                event.connection.send(msg)
```

```
msg = of.ofp_flow_mod()
    msg.priority =10
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.dl_type = 0x0800
    msg.match.nw_src="10.0.0.3"
    msg.match.nw_dst = "10.0.0.1"
    msg.actions.append(of.ofp_action_output(port = 2))
    event.connection.send(msg)

    msg = of.ofp_flow_mod()
    msg.priority =10
    msg.idle_timeout = 0
    msg.hard_timeout = 0
    msg.match.dl_type = 0x0800
    msg.match.nw_src="10.0.0.2"
    msg.match.nw_dst = "10.0.0.1"
    msg.actions.append(of.ofp_action_output(port = 1))
    event.connection.send(msg)
```

```
def launch ():
  core.openflow.addListenerByName("ConnectionUp",
_handle_ConnectionUp)
  core.openflow.addListenerByName("PacketIn",
_handle_PacketIn)
```

```
mininet@mininet-vm:~/mylab$ sudo ./lab4.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s0 s1 s2 s3
*** Adding links:
(10.00Mbit 10ms delay 0% loss) (10.00Mbit 10ms delay 0% loss) (h1, s0) (10.00Mbi
t 10ms delay 0% loss) (10.00Mbit 10ms delay 0% loss) (h2, s3) (10.00Mbit 10ms de
lay 0% loss) (10.00Mbit 10ms delay 0% loss) (h3, s3) (10.00Mbit 10ms delay 50% l
oss) (10.00Mbit 10ms delay 50% loss) (s0, s1) (10.00Mbit 10ms delay 10% loss) (1
0.00Mbit 10ms delay 10% loss) (s0, s2) (10.00Mbit 10ms delay 0% loss) (10.00Mbit
 10ms delay 0% loss) (s1, s3) (10.00Mbit 10ms delay 0% loss) (10.00Mbit 10ms del
ay 0% loss) (s2, s3)
*** Configuring hosts
h1 (cfs 16666/100000us) h2 (cfs 16666/100000us) h3 (cfs 16666/100000us)
*** Starting controller
*** Starting 4 switches
s0 (10.00Mbit 10ms delay 0% loss) (10.00Mbit 10ms delay 50% loss) (10.00Mbit 10m
s delay 10% loss) s1 (10.00Mbit 10ms delay 50% loss) (10.00Mbit 10ms delay 0% lo
ss) s2 (10.00Mbit 10ms delay 10% loss) (10.00Mbit 10ms delay 0% loss) s3 (10.00M
bit 10ms delay 0% loss) (10.00Mbit 10ms delay 0% loss) (10.00Mbit 10ms delay 0%
loss) (10.00Mbit 10ms delay 0% loss)
```

```
Dumping host connections
h1 h1-eth0:s0-eth1
h2 h2-eth0:s3-eth3
h3 h3-eth0:s3-eth4
*** Starting CLI:
mininet> h1 ping -c 20 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_req=1 ttl=64 time=171 ms
64 bytes from 10.0.0.3: icmp_req=3 ttl=64 time=83.9 ms
64 bytes from 10.0.0.3: icmp_req=4 ttl=64 time=83.4 ms
64 bytes from 10.0.0.3: icmp_req=5 ttl=64 time=84.0 ms
64 bytes from 10.0.0.3: icmp_req=6 ttl=64 time=83.3 ms
64 bytes from 10.0.0.3: icmp_req=7 ttl=64 time=83.9 ms
64 bytes from 10.0.0.3: icmp_req=8 ttl=64 time=83.4 ms
64 bytes from 10.0.0.3: icmp_req=9 ttl=64 time=83.9 ms
64 bytes from 10.0.0.3: icmp_req=11 ttl=64 time=83.9 ms
64 bytes from 10.0.0.3: icmp_req=12 ttl=64 time=82.8 ms
64 bytes from 10.0.0.3: icmp_req=14 ttl=64 time=83.3 ms
64 bytes from 10.0.0.3: icmp_req=15 ttl=64 time=83.9 ms
64 bytes from 10.0.0.3: icmp_req=16 ttl=64 time=83.4 ms
64 bytes from 10.0.0.3: icmp_req=17 ttl=64 time=83.9 ms
64 bytes from 10.0.0.3: icmp_req=18 ttl=64 time=83.3 ms
64 bytes from 10.0.0.3: icmp_req=19 ttl=64 time=83.9 ms
64 bytes from 10.0.0.3: icmp_req=20 ttl=64 time=83.4 ms

--- 10.0.0.3 ping statistics ---
20 packets transmitted, 17 received, 15% packet loss, time 19034ms
rtt min/avg/max/mdev = 82.874/88.823/171.449/20.662 ms
```

```
mininet> h1 ping -c 20 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_req=2 ttl=64 time=83.4 ms
64 bytes from 10.0.0.2: icmp_req=3 ttl=64 time=83.4 ms
64 bytes from 10.0.0.2: icmp_req=5 ttl=64 time=83.2 ms
64 bytes from 10.0.0.2: icmp_req=7 ttl=64 time=83.8 ms
64 bytes from 10.0.0.2: icmp_req=11 ttl=64 time=98.3 ms
64 bytes from 10.0.0.2: icmp_req=15 ttl=64 time=91.1 ms
64 bytes from 10.0.0.2: icmp_req=20 ttl=64 time=91.9 ms

--- 10.0.0.2 ping statistics ---
20 packets transmitted, 7 received, 65% packet loss, time 19040ms
rtt min/avg/max/mdev = 83.297/87.923/98.331/5.513 ms
mininet> exit
*** Stopping 4 switches
s0 ...s1 ..s2 ..s3 ....
*** Stopping 3 hosts
h1 h2 h3
*** Stopping 1 controllers
c0
*** Done
```

References

- http://mininet.org/
- http://eventos.redclara.net/indico/getFile.py/access?contribId=1&resId=3&materialId=slides&confId=197
- https://github.com/mininet/mininet/wiki/Introduction-to-Mininet
- https://openflow.stanford.edu/display/ONL/POX+Wiki