Lab 9 (Traffic Measurement)

In this lab, the number of packets or bytes for IP or ARP will be recorded in Table 0. Then IP traffic will be forwarded to Table 5 for further classification. The number of packets or bytes for ICMP, TCP, or UDP will be recorded. Finally, all packets will be forwarded to table 10 for flooding.

Put this file (measure_traffic.py) under /pox/ext

```
from pox.core import core
from pox.lib.addresses import EthAddr
import pox.openflow.libopenflow 01 as of
import pox.openflow.nicira as nx
from pox.lib.revent import EventRemove
from pox.lib.packet.arp import arp
from pox.lib.packet.ipv4 import ipv4
import pox.lib.packet as pkt
# Even a simple usage of the logger is much nicer than print!
log = core.getLogger()
def _handle_ConnectionUp (event):
 print " handle ConnectionUP"
 # Set up this switch.
 # After setting up, we send a barrier and wait for the response
 # before starting to listen to packet ins for this switch -- before
 # the switch is set up, the packet ins may not be what we expect,
 # and our responses may not work!
 # Turn on Nicira packet ins
 msg = nx.nx packet in format()
 event.connection.send(msg)
# Turn on ability to specify table in flow mods
 msg = nx.nx flow mod table id()
 event.connection.send(msg)
```

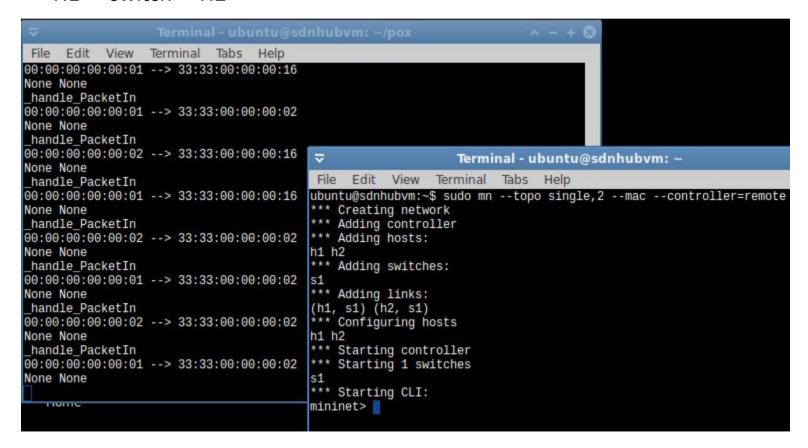
```
# Fallthrough rule for table 0: flood and send to controller
msg = nx.nx flow mod()
msg.priority = 1 # Low priority
msg.actions.append(of.ofp action output(port = of.OFPP CONTROLLER))
msg.actions.append(nx.nx action resubmit.resubmit table(table = 10))
event.connection.send(msg)
# Fallthrough rule for table 1: flood
msg = nx.nx flow mod()
msg.table id = 10
msg.priority = 1 # Low priority
msg.actions.append(of.ofp action output(port = of.OFPP FLOOD))
event.connection.send(msg)
def ready (event):
if event.ofp.xid != 0x80000000:
  # Not the right barrier
  return
log.info("%s ready", event.connection)
 event.connection.addListenerByName("PacketIn", handle PacketIn)
 return EventRemove
event.connection.send(of.ofp_barrier_request(xid=0x80000000))
event.connection.addListenerByName("BarrierIn", ready)
```

```
def handle PacketIn(event):
 print " handle PacketIn"
 packet = event.parsed
 if event.port > of.OFPP MAX:
 log.debug("Ignoring special port %s", event.port)
 return
 print packet.src, "-->", packet.dst
 a=packet.find('arp')
 b=packet.find('ipv4')
 print a, b
 #print "ICMP Packet"
 msg = nx.nx flow mod()
 msg.table id = 5
 msg.match.of eth type = pkt.ethernet.IP TYPE
 msg.match.of_ip_proto = 1
 msg.actions.append(nx.nx action resubmit.resubmit table(table = 10))
 event.connection.send(msg)
 #print "TCP Packet"
 msg = nx.nx flow mod()
 msg.table id = 5
 msg.match.of_eth_type = pkt.ethernet.IP_TYPE
 msg.match.of ip proto = 6
 msg.actions.append(nx.nx action resubmit.resubmit table(table = 10))
 event.connection.send(msg)
```

```
#print "UDP Packet"
 msg = nx.nx flow mod()
 msg.table id = 5
 msg.match.of_eth_type = pkt.ethernet.IP_TYPE
 msg.match.of ip proto = 17
 msg.actions.append(nx.nx_action_resubmit.resubmit_table(table = 10))
 event.connection.send(msg)
 #print "ARP Packet"
 msg = of.ofp flow mod()
 msg.priority =100
 msg.match.dl type = 0x0806
 msg.actions.append(nx.nx action resubmit.resubmit table(table = 10))
 event.connection.send(msg)
 #print "IP Packet"
 msg = of.ofp flow mod()
 msg.priority =100
 msg.match.dl_type = 0x0800
 msg.actions.append(nx.nx action resubmit.resubmit table(table = 5))
 event.connection.send(msg)
def launch ():
 assert core.NX, "Nicira extensions required"
 assert core.NX.convert packet in, "PacketIn conversion required"
 core.openflow.addListenerByName("ConnectionUp", handle ConnectionUp)
 log.info("Simple NX switch running.")
```

```
ubuntu@sdnhubvm:~/pox$ ./pox.py openflow.nicira --convert-packet-in measure_traf
fic
POX 0.1.0 (betta) / Copyright 2011-2013 James McCauley, et al.
INFO:measure_traffic:Simple NX switch running.
INFO:core:POX 0.1.0 (betta) is up.
```

H1----switch ---H2



Use dpctl command to directly check the information in switch

```
mininet> s1 dpctl dump-flows tcp:12/.0.0.1:6634
stats reply (xid=0xe86f14b5): flags=none type=1(flow)
  cookie=0, duration sec=95s, duration nsec=734000000s, table id=0, priority=1,
n packets=14, n bytes=1116, idle timeout=0, hard timeout=0, actions=CONTROLLER:655
35, vendor action:0x2320
  cookie=0, duration sec=86s, duration nsec=654000000s, table id=0, priority=100
, n packets=0, n bytes=0, idle timeout=0, hard timeout=0, ip, actions=vendor action
:0x2320
  cookie=0, duration sec=86s, duration nsec=655000000s, table id=0, priority=100
 n packets=0, n bytes=0, idle timeout=0, hard timeout=0, arp, actions=vendor actio
n:0x2320
  cookie=0, duration_sec=86s, duration_nsec=655000000s, table_id=5, priority=327
68, n packets=0, n bytes=0, idle timeout=0, hard timeout=0, icmp, actions=vendor ac
tion:0x2320
  cookie=0, duration_sec=86s, duration_nsec=655000000s, table_id=5, priority=327
68, n packets=0, n bytes=0, idle timeout=0,hard timeout=0,udp,actions=vendor act
ion:0x2320
  cookie=0, duration sec=86s, duration nsec=655000000s, table_id=5, priority=327
68, n packets=0, n bytes=0, idle timeout=0,hard timeout=0,tcp,actions=vendor act
ion:0x2320
  cookie=0, duration sec=95s, duration nsec=734000000s, table id=10, priority=1,
n packets=14, n bytes=1116, idle timeout=0, hard timeout=0, actions=FLOOD
mininet>
```

Initially, no ip, arp, icmp, udp, and tcp traffic.

```
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_seq=1 ttl=64 time=1.06 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.110 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.093 ms
--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.093/0.421/1.062/0.453 ms
```

After executing ping from h1 to h2,

```
mininet> s1 dpctl dump-flows tcp:127.0.0.1:6634
stats reply (xid=0x8de5d838): flags=none type=1(flow)
 cookie=0, duration sec=284s, duration nsec=383000000s, table id=0, priority=1
n_packets=14, n_bytes=1116, idle_timeout=0,hard_timeout=0,actions=CONTROLLER:65
535, vendor action:0x2320
 cookie=0, duration_sec=275s, duration_nsec=303000000s, table_id=0, priority=10

    n packets=6, n bytes=588, idle timeout=0, hard timeout=0, ip, actions=vendor act

ion:0x2320
 cookie=0, duration_sec=275s, duration_nsec=304000000s, table id=0, priority=10
n_packets=4, n_bytes=168, idle_timeout=0,hard_timeout=0,arp,actions=vendor ac
tion:0x2320
 cookie=0, duration_sec=275s, duration_nsec=304000000s, table_id=5, priority=32
768, n packets=6, n bytes=588, idle_timeout=0,hard_timeout=0,icmp,actions=vendor
action:0x2320
 cookie=0, duration_sec=275s, duration_nsec=304000000s, table id=5, priority=32
768, n_packets=0, n_bytes=0, idle_timeout=0,hard_timeout=0,udp,actions=vendor_ac
tion:0x2320
 cookie=0, duration_sec=275s, duration_nsec=304000000s, table_id=5, priority=32
768, n_packets=0, n_bytes=0, idle_timeout=0,hard_timeout=0,tcp,actions=vendor_ac
tion:0x2320
 cookie=0, duration_sec=284s, duration_nsec=383000000s, table_id=10, priority=1
 n packets=24, n bytes=1872, idle timeout=0,hard timeout=0,actions=FLOOD
```

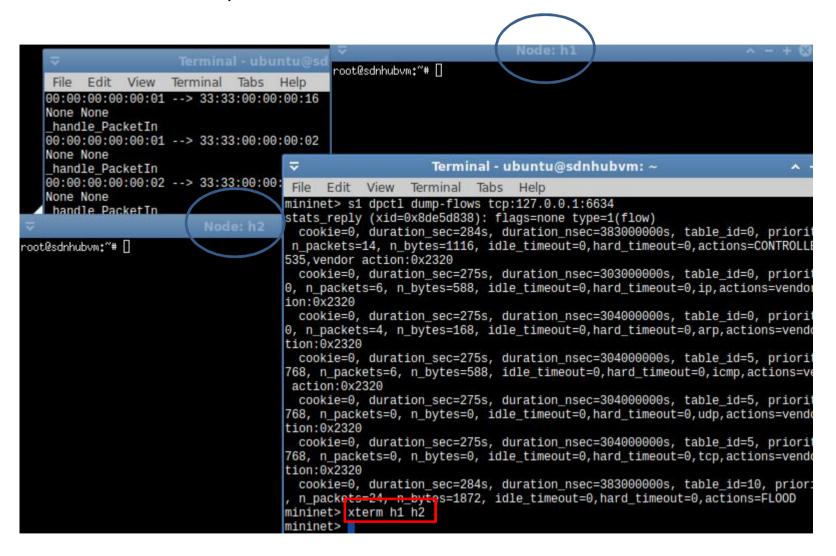
IP packets: 6 packets (3 ping request packets and 3 ping response packets.)

ARP packet: 4 packets

ICMP packets: 6 packets (3 ping request packets and 3 ping response packets.)

No UDP nor TCP traffic

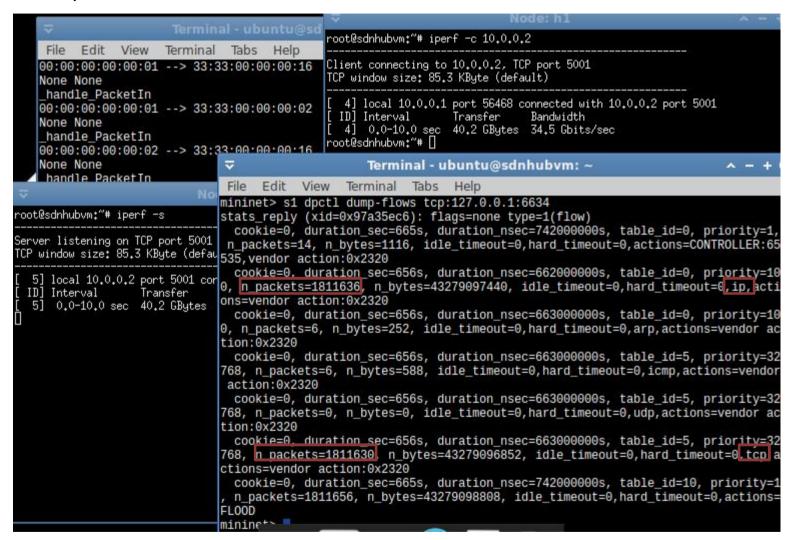
Open terminals for h1 and h2



h1: iperf -c 10.0.0.2

h2: iperf -s

TCP traffic



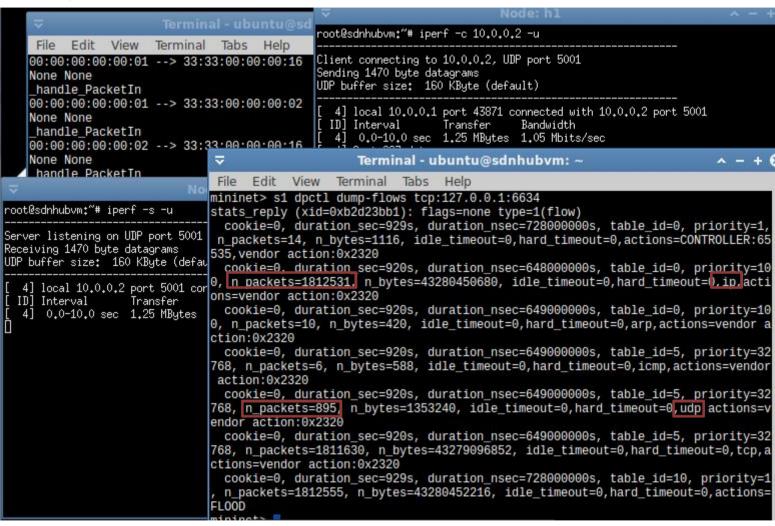
IP: 1811636 (TCP:1811630 packets + ICMP: 6 packets)

TCP: 1811630 packets

h1: iperf -c 10.0.0.2 -u

h2: iperf –s -u

UDP traffic



IP: 1812531 packets(UDP: 895 TCP:1811630 packets + ICMP: 6 packets)

UDP: 895 packets

References

• POX Wiki, Nicira/Open vSwitch Extensions https://openflow.stanford.edu/display/ONL/POX+Wiki

Dr. Chih-Heng Ke

http://csie.nqu.edu.tw/smallko

SDN website: http://csie.nqu.edu.tw/smallko/sdn/sdn.htm