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.
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)

Put this file (measure_traffic.py) under /pox/ext
# 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.py openflow.nicira --convert-packet-in measure_traffic
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.

ubuntux@sdnhubvm:~$ sudo mn --topo single,2 --mac --controller=remote
*** Creating network
*** Adding Controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links: (h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
*** Starting 1 switches
s1
*** Starting CLI:
mininet>
Use dpctl command to directly check the information in switch

Initially, no ip, arp, icmp, udp, and tcp traffic.
After executing ping from h1 to h2,

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`

UDP traffic

h2: `iperf -s -u`

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