Lab 3

```
mininet@mininet-vm:~$ sudo ~/pox/pox.py misc.lab3controller
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:[None 1] closed
WARNING:openflow.of_01:<class 'pox.openflow.PortStatus'> raised on dummy OpenFl
w nexus
WARNING:openflow.of_01:<class 'pox.openflow.PortStatus'> raised on dummy OpenFl
w nexus
WARNING:openflow.of_01:<class 'pox.openflow.PortStatus'> raised on dummy OpenFl
w nexus
WARNING:openflow.of_01:<class 'pox.openflow.PortStatus'> raised on dummy OpenFl
w nexus
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
```

We start by running “sudo ~/pox/pox.py misc.lab3controller”, followed by “sudo python ~/lab3.py”.

pingall:

```
mininet@mininet-vm:~$ sudo python ~/lab3.py
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X X X
h3 -> X X X
h4 -> X X X
*** Results: 100% dropped (0/12 received)
```

When running the pingall command in Mininet, it sends ICMP echo request packets from each host to every other host in the network. Since ICMP traffic is blocked by the firewall, the pingall command fails, resulting in 12 drops.

dpctl dump-flows:

```
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=31.937s, table=0, n_packets=0, n_bytes=0, idle_age=31, pr
ority=9,tcp actions=FLOOD
 cookie=0x0, duration=31.926s, table=0, n_packets=0, n_bytes=0, idle_age=31, pr
ority=1,ip actions=drop
 cookie=0x0, duration=31.974s, table=0, n_packets=0, n_bytes=0, idle_age=31, pr
ority=10,arp actions=FLOOD
```

Running dpctl dump-flows in the Mininet CLI displays the flow entries installed in the OpenFlow switch by the controller. These flow entries correspond to the firewall rules installed by the controller using ofp_flow_mod.

iperf:

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['5.11 Gbits/sec', '5.12 Gbits/sec']
```

Running iperf between two hosts in the Mininet network tests the network's performance by measuring the throughput between the hosts. A successful iperf test indicates that traffic is able to flow freely between the hosts without being blocked by the firewall.

CODE:

```
from pox.core import core
import pox.openflow.libopenflow_01 as of
log = core.getLogger()
class Firewall (object):
  def __init__ (self, connection):
    self.connection = connection
    connection.addListeners(self)
    self.install_rules()
  def install_rules(self):
    msg = of.ofp_flow_mod()
    msg.priority = 10
    msg.match.dl_type = 0x0806
    msg.actions.append(of.ofp_action_output(port=of.OFPP_FLOOD))
    self.connection.send(msg)
    msg = of.ofp_flow_mod()
    msg.priority = 9
    msg.match.dl_type = 0x0800
    msg.match.nw_proto = 6
    msg.actions.append(of.ofp_action_output(port=of.OFPP_FLOOD))
```

```
    self.connection.send(msg)
    msg = of.ofp_flow_mod()
    msg.priority = 1
    msg.match.dl_type = 0x0800
    self.connection.send(msg)
  def do_firewall(self, packet, packet_in):
    if packet.type == packet.ARP_TYPE:
        log.debug("Allowing and flooding ARP packet")
        self.flood_packet(packet_in)
    elif packet.type == packet.IP_TYPE:
        ip_packet = packet.payload
        if ip_packet.protocol == 6:
            log.debug("Allowing and flooding TCP packet")
            self.flood_packet(packet_in)
        else:
            log.debug("Dropping non-TCP IPv4 packet")
    else:
        log.debug("Dropping packet of type %s" % packet.type)
  def flood_packet(self, packet_in):
    msg = of.ofp_packet_out()
    msg.data = packet_in
    action = of.ofp_action_output(port=of.OFPP_FLOOD)
    msg.actions.append(action)
    self.connection.send(msg)
  def _handle_PacketIn (self, event):
    packet = event.parsed
    if not packet.parsed:
      log.warning("Ignoring incomplete packet")
      return
    packet_in = event.ofp
    self.do_firewall(packet, packet_in)
def launch ():
  def start_switch (event):
    log.debug("Controlling %s" % (event.connection,))
    Firewall(event.connection)
  core.openflow.addListenerByName("ConnectionUp", start_switch)
```