CO515: Advances in Computer Networks: Selected Topics – Lab07 (Mininet Lab 04)

E/19/446: Wijerathna I.M.K.D.I.

16/05/2024

Understanding Packet Routing in Mininet OpenFlow

Install Mininet and Pox

```
vagrant@sdn-box:~$ sudo apt-get update
Ign http://archive.ubuntu.com trusty InRelease
Get:1 http://security.ubuntu.com trusty-security InRelease [56.4 kB]
Get:2 http://archive.ubuntu.com trusty-updates InRelease [56.5 kB]
Hit http://archive.ubuntu.com trusty Release.gpg
Get:3 http://security.ubuntu.com trusty-security/main Sources [254 kB]
Get:4 http://archive.ubuntu.com trusty-updates/main Sources [667 kB]
Get:5 http://security.ubuntu.com trusty-security/universe Sources [154 kB]
Get:6 http://archive.ubuntu.com trusty-updates/universe Sources [356 kB]
Get:7 http://security.ubuntu.com trusty-security/main amd64 Packages [702 kB]
Get:8 http://archive.ubuntu.com trusty-updates/main amd64 Packages [1,172 kB]
```

Figure 1: Updating the OS

```
vagrant@sdn-box:~$ sudo apt-get install -y mininet
Reading package lists... Done
Building dependency tree
Reading state information... Done
Recommended packages:
  openvswitch-controller
The following NEW packages will be installed:
  mininet
```

Figure 2: Installation of Mininet

```
vagrant@sdn-box:~$ cd pox
vagrant@sdn-box:~/pox$ ls
debug-pox.py  ext  LICENSE  NOTICE  pox  pox.py  README  setup.cfg  tests  tools
vagrant@sdn-box:~/pox$
```

Figure 3: Ensuring the Download of the Pox Repository

Lab Environment Setup

I used the same setup as the previous labs (Mininet Lab1,2,3) which is a lightweight ubuntu os setup which runs over vagrant+virtual box. The following is the lab environment structure.

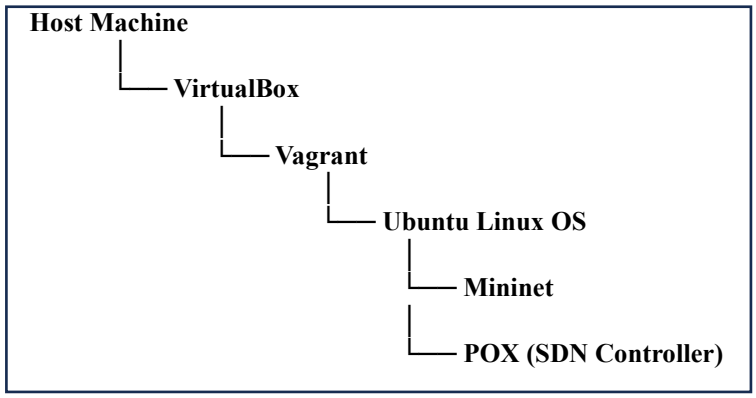

Figure 4: Lab Environment Setup

Create a Basic Mininet Topology (1 switch and 3 hosts)

```
vagrant@sdn-box:~$ sudo mn --topo single,3 --controller=remote --switch ovsk,protocols=OpenFlow10
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

Figure 5: Basic Topology with a remote SDN Controller

```
mininet>  h1 ifconfig h1-eth0 10.0.0.1 netmask 255.255.255.0
mininet>  h2 ifconfig h2-eth0 10.0.0.2 netmask 255.255.255.0
mininet>  h3 ifconfig h3-eth0 10.0.0.3 netmask 255.255.255.0
```

Figure 6: IP Configuration on Hosts

According to the obtained details the following should be the topolgy.

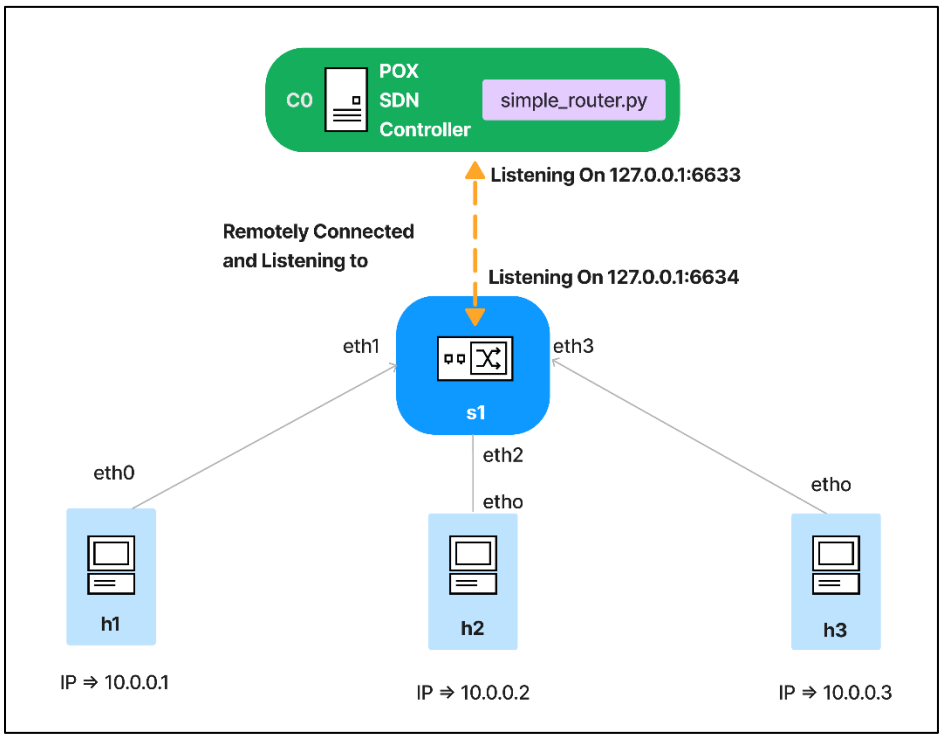

Figure 7: The Mininet Topology

Testting Connectivity before Running the Pox Controller

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X X
h3 -> X X
*** Results: 100% dropped (0/6 received)
```

Figure 8: pingall before connecting the Pox Controller

Since, No flow rules have been configured so far; none of the hosts could communicate with each other.

Creating a Simple Router with Static Routing Logic

The following is the Python script for simple_router.py

```python
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.util import dpid_to_str
from pox.lib.addresses import IPAddr, EthAddr
from pox.lib.packet.ethernet import ethernet
from pox.lib.packet.ipv4 import ipv4
from pox.lib.packet.arp import arp


# Get a logger for debugging
log = core.getLogger()

class SimpleRouter(object):
    def __init__(self, connection):
        self.connection = connection  # Save the connection to the switch
        connection.addListeners(self)  # Listen for events on this connection


    def _handle_PacketIn(self, event):
        """
        This function is called whenever a packet is received from the switch.
        """
        packet = event.parsed  # The parsed packet data
        in_port = event.port  # The port on which the packet was received

        if not packet.parsed:
            log.warning("Ignoring incomplete packet")
            return

        # Handle ARP packets
        if packet.type == ethernet.ARP_TYPE:
            log.debug("Handling ARP packet")
            self._handle_arp(packet, in_port, event.ofp)
        # Handle IP packets
        elif packet.type == ethernet.IP_TYPE:
            log.debug("Handling IP packet")
            self._handle_static_routes(packet, in_port, event.ofp)
        else:
            log.debug("Ignoring non-IP/ARP packet type: %s", packet.type)
```

```
    def _handle_arp(self, packet, in_port, ofp):

         '''  Handle ARP packets
     '''
        arp_packet = packet.payload
        if arp_packet.opcode == arp.REQUEST or arp_packet.opcode == arp.REPLY:
            log.debug("Flooding ARP packet: %s", arp_packet)

            ether = ethernet()
            ether.type = ethernet.ARP_TYPE
            ether.dst = EthAddr("ff:ff:ff:ff:ff:ff")  # Broadcast MAC address
            ether.src = packet.src  # Use the source MAC address from the packet
            ether.payload = arp_packet

            msg = of.ofp_packet_out()
            msg.data = ether.pack()
            msg.actions.append(of.ofp_action_output(port=of.OFPP_FLOOD))
            msg.in_port = in_port
            self.connection.send(msg)



    def _handle_static_routes(self, packet, in_port, ofp):

    '''  Handle Ip packets with static routing configured below
     '''
        ipv4_packet = packet.payload  # Extract the IP payload
        ipv4_dst_ip = ipv4_packet.dstip  # Get the destination IP address from the IP packet

        log.debug("IP packet recieved with destined to: %s.", ipv4_dst_ip)

    #static routes for hosts h1,h2.h3

        if(ipv4_dst_ip == '10.0.0.1'):
         out_port = 1

        elif(ipv4_dst_ip == '10.0.0.2'):
            out_port = 2

        elif(ipv4_dst_ip == '10.0.0.3'):
            out_port = 3

        else:
            log.debug("Unknown destination IP")
            return

        # Create actions to forward the packet to the determined port
        actions = [of.ofp_action_output(port=out_port)]
        # Create a flow match based on the received packet
        match = of.ofp_match.from_packet(packet, in_port)
        # Create a flow mod message to install the flow rule
        msg = of.ofp_flow_mod()
        msg.match = match
        msg.idle_timeout = 10  # Flow idle timeout
        msg.hard_timeout = 30  # Flow hard timeout
        msg.actions = actions
        msg.data = ofp  # Include the original packet in the message
        self.connection.send(msg)

        log.debug("Flow installed for %s to %s out port %s", ipv4_packet.srcip, ipv4_dst_ip,
out_port)
```

```
def launch():
    """
    Start the SimpleRouter component.
     """
    def start_switch(event):
        log.debug("Controlling %s" % (dpid_to_str(event.dpid)))
        SimpleRouter(event.connection)  # Create an instance of SimpleRouter for each switch
connection
    core.openflow.addListenerByName("ConnectionUp", start_switch)  # Add a listener for switch
connections
```

Run the POX controller

Creating the simple_router.py file inside the correct directory under ./pox and starting the Pox controller

```
vagrant@sdn-box:~$ cd pox/
vagrant@sdn-box:~/pox$ sudo vim ./pox/forwarding/simple_router.py
vagrant@sdn-box:~/pox$ ./pox.py log.level --DEBUG forwarding.simple_router
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-3.13.0-49-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
DEBUG:forwarding.simple_router:Controlling 00-00-00-00-00-01
```

Figure 9: Starting the Pox SDN Controller with simple_router.py

Right-after the pox controller started it was observed at the port 6633 from wireshark, OpenFlow packets for initializing the connection between the pox controller (c0) and the OpenFlow switch (s1) have been exchanged.

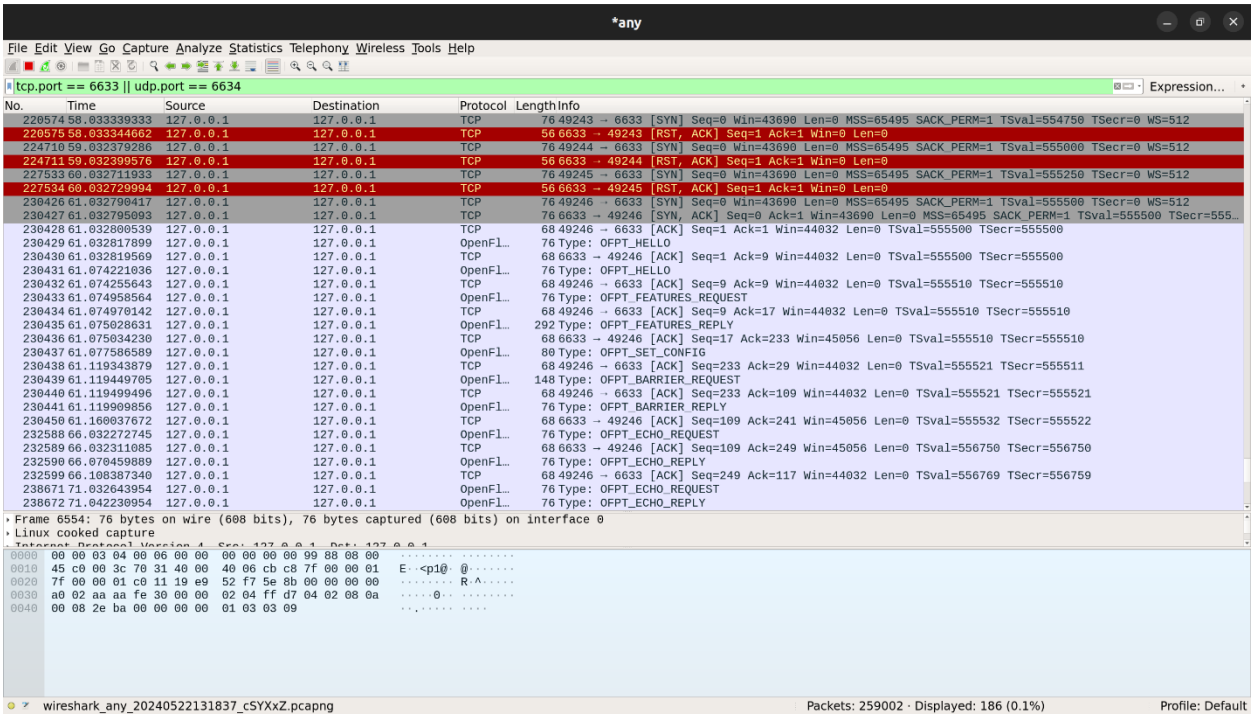

Figure 10: Wireshark Snapshot When Starting the Pox Controller

Testing Connectivity

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet>
```

Figure 11: pingall after Starting the Pox controller with static router logic

Debug terminal of the pox controller:

```
vagrant@sdn-box:~/pox$ ./pox.py log.level --DEBUG forwarding.simple_router
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-3.13.0-49-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
DEBUG:forwarding.simple_router:Controlling 00-00-00-00-00-01
DEBUG:forwarding.simple_router:Handling ARP packet
DEBUG:forwarding.simple_router:Flooding ARP packet: [ARP REQUEST hw:1 p:2048 b2:0a:58:b0:d7:41>00:00:00:00:00:00 10.0.0.1>10.0.0.2]
DEBUG:forwarding.simple_router:Handling ARP packet
DEBUG:forwarding.simple_router:Flooding ARP packet: [ARP REPLY hw:1 p:2048 22:15:4a:05:3e:68>b2:0a:58:b0:d7:41 10.0.0.2>10.0.0.1]
DEBUG:forwarding.simple_router:Handling IP packet
DEBUG:forwarding.simple_router:IP packet recieved with destined to: 10.0.0.1.
DEBUG:forwarding.simple_router:Flow installed for 10.0.0.2 to 10.0.0.1 out port 1
DEBUG:forwarding.simple_router:Handling ARP packet
DEBUG:forwarding.simple_router:Flooding ARP packet: [ARP REQUEST hw:1 p:2048 b2:0a:58:b0:d7:41>00:00:00:00:00:00 10.0.0.1>10.0.0.3]
DEBUG:forwarding.simple_router:Handling ARP packet
DEBUG:forwarding.simple_router:Flooding ARP packet: [ARP REPLY hw:1 p:2048 36:9b:3f:a9:db:e5>b2:0a:58:b0:d7:41 10.0.0.3>10.0.0.1]
DEBUG:forwarding.simple_router:Handling IP packet
DEBUG:forwarding.simple_router:IP packet recieved with destined to: 10.0.0.3.
DEBUG:forwarding.simple_router:Flow installed for 10.0.0.1 to 10.0.0.3 out port 3
DEBUG:forwarding.simple_router:Handling IP packet
DEBUG:forwarding.simple_router:IP packet recieved with destined to: 10.0.0.1.
DEBUG:forwarding.simple_router:Flow installed for 10.0.0.3 to 10.0.0.1 out port 1
DEBUG:forwarding.simple_router:Handling IP packet
DEBUG:forwarding.simple_router:IP packet recieved with destined to: 10.0.0.1.
DEBUG:forwarding.simple_router:Flow installed for 10.0.0.2 to 10.0.0.1 out port 1
DEBUG:forwarding.simple_router:Handling IP packet
DEBUG:forwarding.simple_router:IP packet recieved with destined to: 10.0.0.2.
DEBUG:forwarding.simple_router:Flow installed for 10.0.0.1 to 10.0.0.2 out port 2
DEBUG:forwarding.simple_router:Handling IP packet
DEBUG:forwarding.simple_router:IP packet recieved with destined to: 10.0.0.3.
DEBUG:forwarding.simple_router:Flow installed for 10.0.0.2 to 10.0.0.3 out port 3
DEBUG:forwarding.simple_router:Handling IP packet
DEBUG:forwarding.simple_router:IP packet recieved with destined to: 10.0.0.2.
DEBUG:forwarding.simple_router:Flow installed for 10.0.0.3 to 10.0.0.2 out port 2
DEBUG:forwarding.simple_router:Handling IP packet
DEBUG:forwarding.simple_router:IP packet recieved with destined to: 10.0.0.1.
DEBUG:forwarding.simple_router:Flow installed for 10.0.0.3 to 10.0.0.1 out port 1
DEBUG:forwarding.simple_router:Handling IP packet
DEBUG:forwarding.simple_router:IP packet recieved with destined to: 10.0.0.3.
DEBUG:forwarding.simple_router:Flow installed for 10.0.0.1 to 10.0.0.3 out port 3
DEBUG:forwarding.simple_router:Handling IP packet
DEBUG:forwarding.simple_router:IP packet recieved with destined to: 10.0.0.2.
DEBUG:forwarding.simple_router:Flow installed for 10.0.0.3 to 10.0.0.2 out port 2
DEBUG:forwarding.simple_router:Handling IP packet
DEBUG:forwarding.simple_router:IP packet recieved with destined to: 10.0.0.3.
DEBUG:forwarding.simple_router:Flow installed for 10.0.0.2 to 10.0.0.3 out port 3
DEBUG:forwarding.simple_router:Handling ARP packet
DEBUG:forwarding.simple_router:Flooding ARP packet: [ARP REQUEST hw:1 p:2048 b2:0a:58:b0:d7:41>00:00:00:00:00:00 10.0.0.1>10.0.0.2]
DEBUG:forwarding.simple_router:Handling ARP packet
```

Figure 12: Debug Terminal of Pox: part-2

The above debug terminal logs from the pox controller, indicates the ARP requests-responses, OpenFlow rules for static routing being matched and routed, and OpenFlow rules being installed to the OpenFlow switch.

To ensure the echo requests-responses, the packets have been captured through wireshark as well.

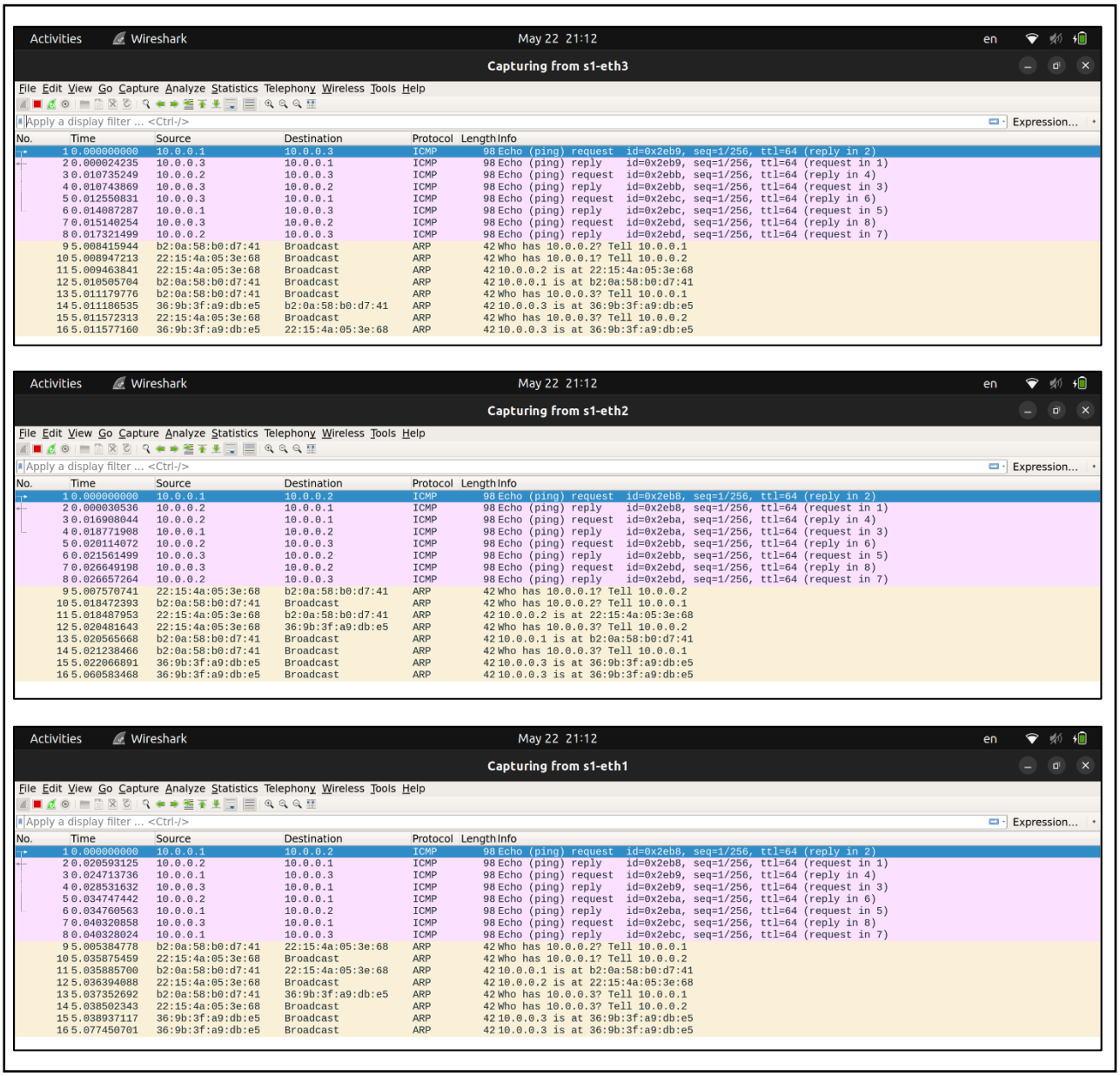

Figure 13: Wireshark Snapshots of Hosts h3, h2, h1 connected switch interfaces: from Top to Bottom Respectively

It could be clearly observed that the echo requests and responses have been exchanged successfully between the hosts during the 'pingall'.

Moreover, during the 'pingall' OFPT_PACKET_IN and OFPT_PACKET_OUT packets were captured at the controller(c0)'s interface port

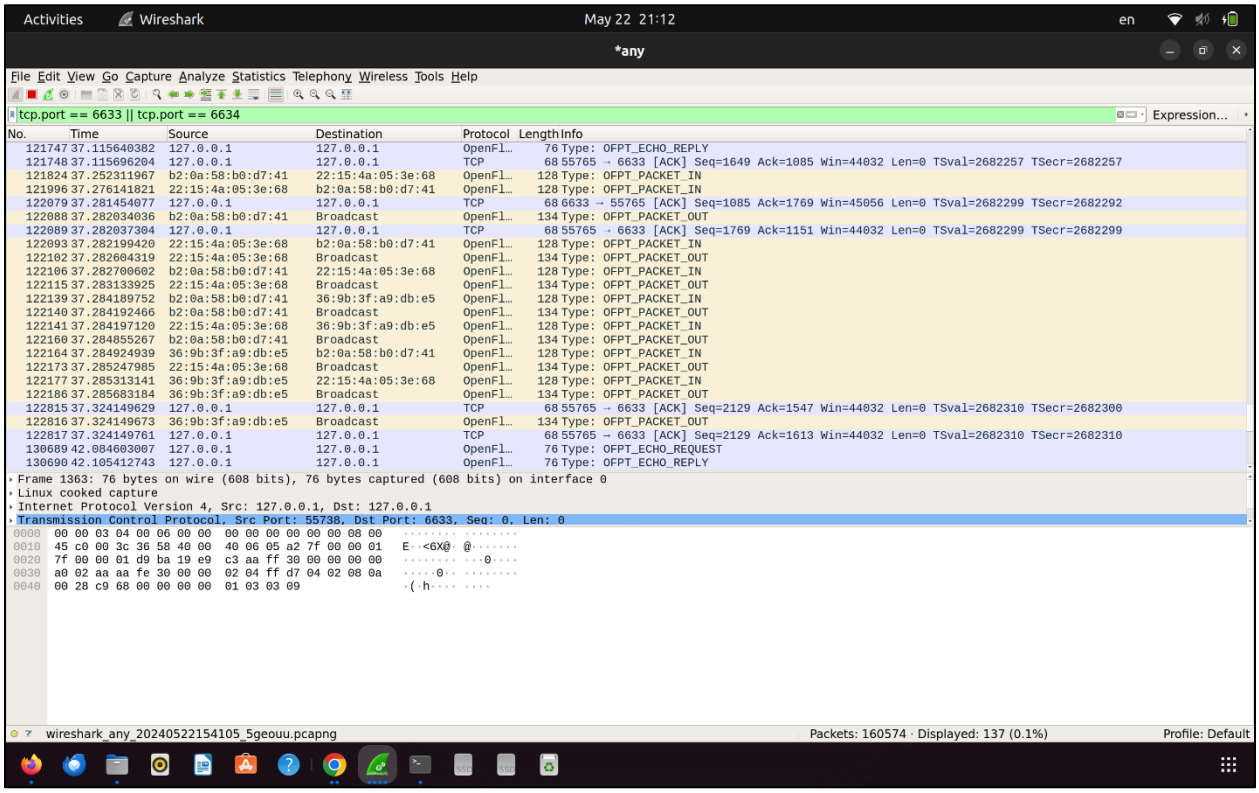

Figure 14: Wirshark Packet Capturing at the link in between switch(s1) and pox controller(c0)

Cleaning-Up the Project

```
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 3 links
...
*** Stopping 1 switches
s1
*** Stopping 3 hosts
h1 h2 h3
*** Done
completed in 4038.761 seconds
vagrant@sdn-box:~$ sudo mn -c
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller udpbwtest mnexec ivs 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd ovs-controller udpbwtest mnexec ivs 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
vagrant@sdn-box:~$
```

Figure 15: Cleaning the Mininet Topology

```
INFO:openflow.of_01:[00-00-00-00-00-01 1] closed

^CINFO:core:Going down...
INFO:core:Down.
vagrant@sdn-box:~/pox$
```

Figure 16: Stopping the Pox Controller with simple_router.py