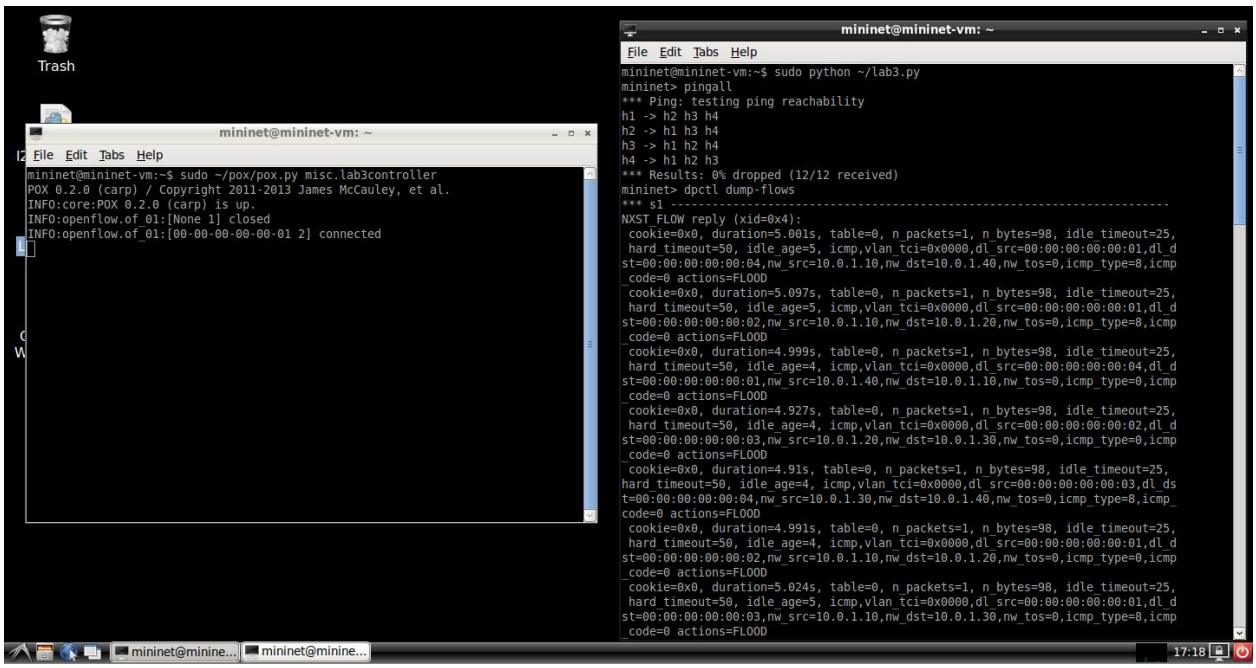
Shawn Chumbar
CMPE 150
Professor Christina Parsa

Lab 3

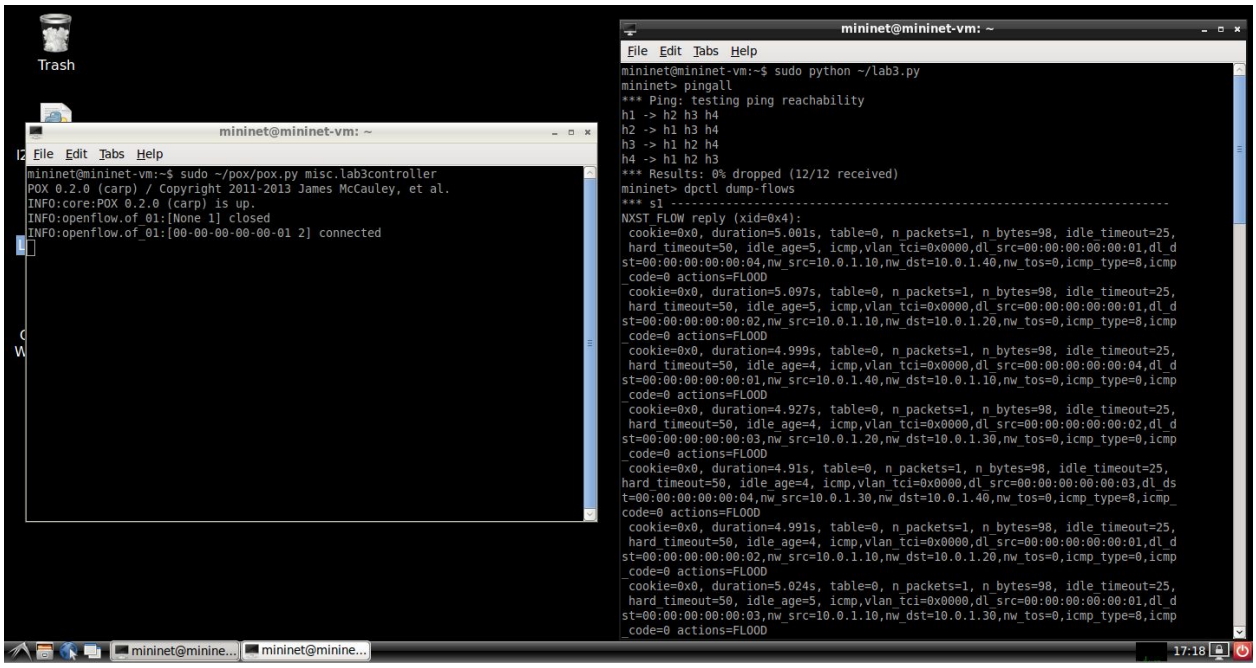
pingall:

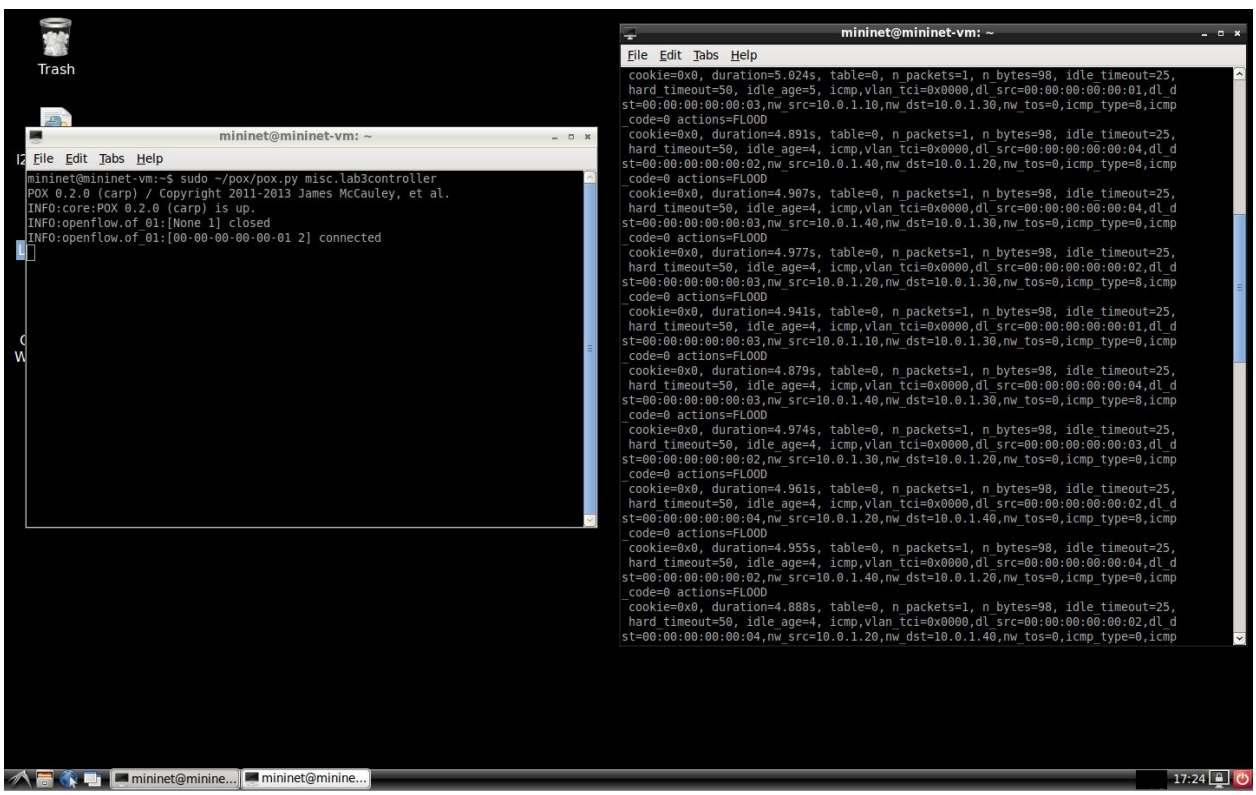
When running the pingall command, we are using ICMP and ARP, which allows the protocol to go through. In this case, the traffic should be allowed to flow through all of the hosts since there is no blockage in place.

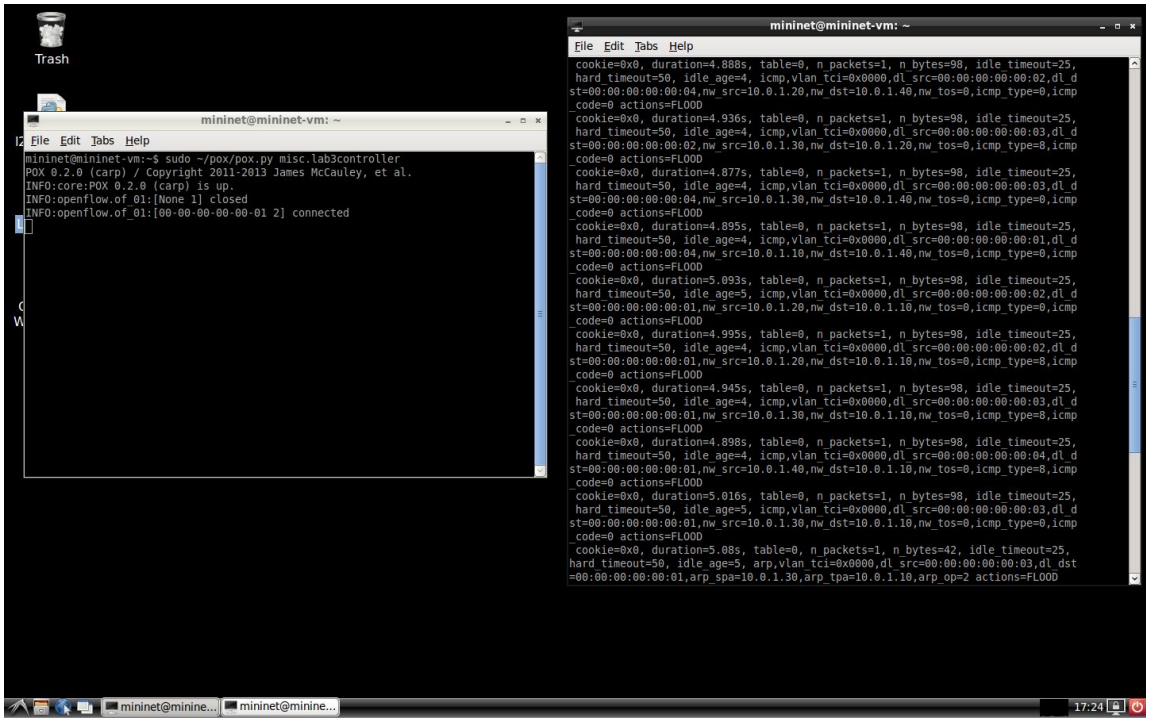


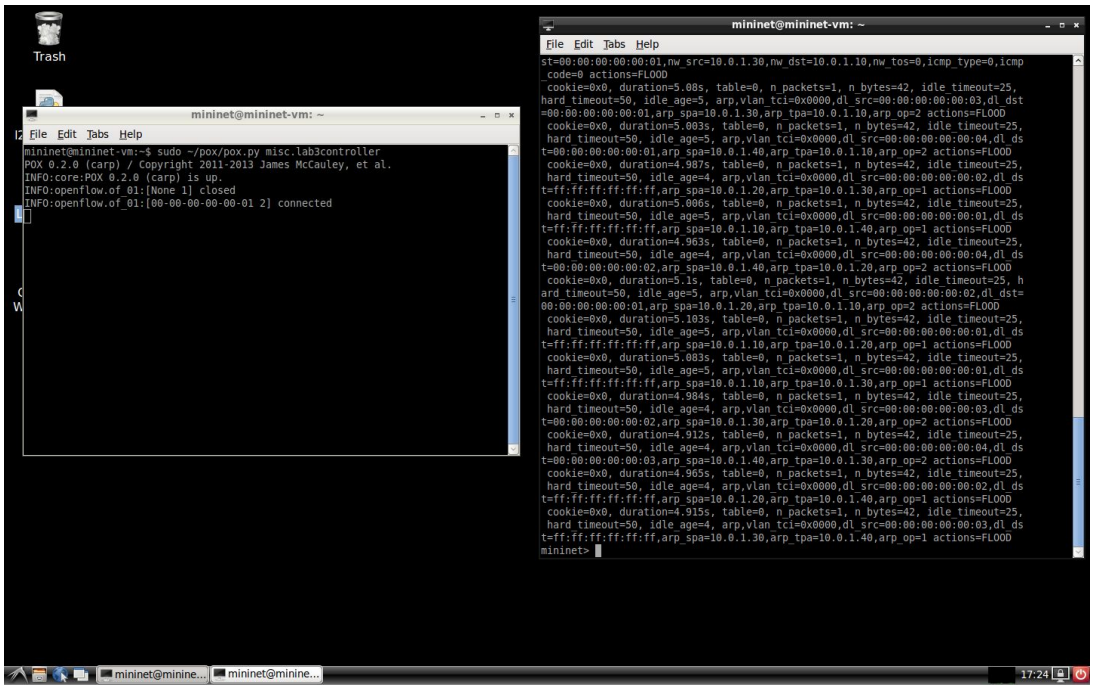
dpctl dump-flows:

This command shows us the type of traffic that pingall used, and the various actions that the controller took. We can see that, for example, in the first entry, we used icmp and it took us 5.001 seconds to transfer the data. The action that was taken was to FLOOD the network.



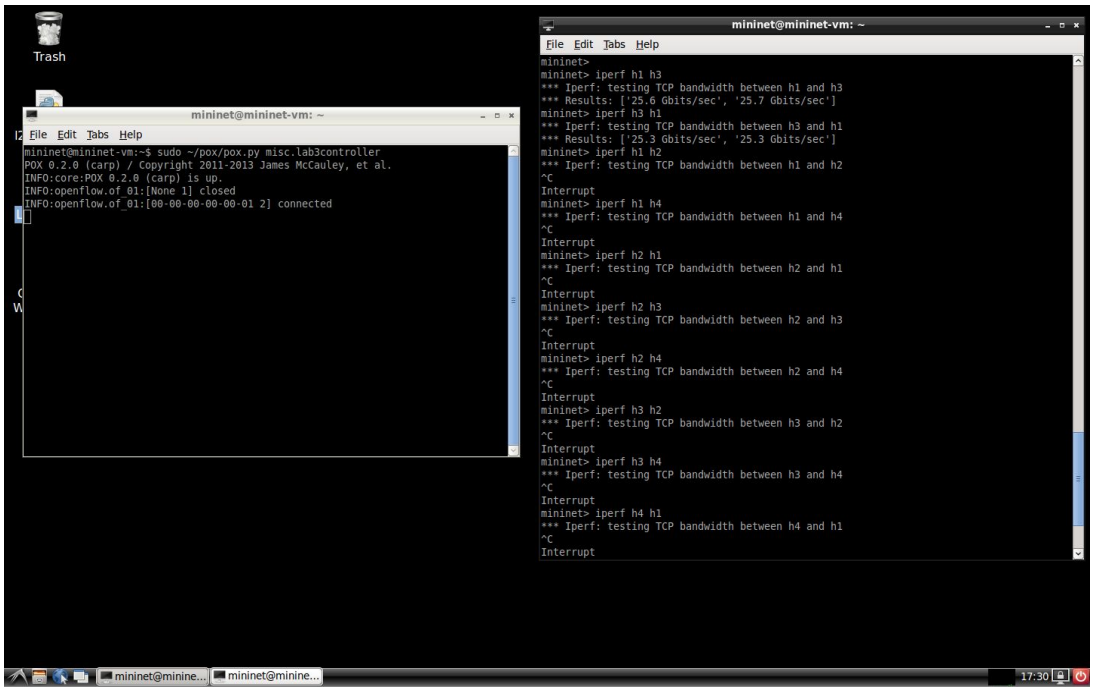


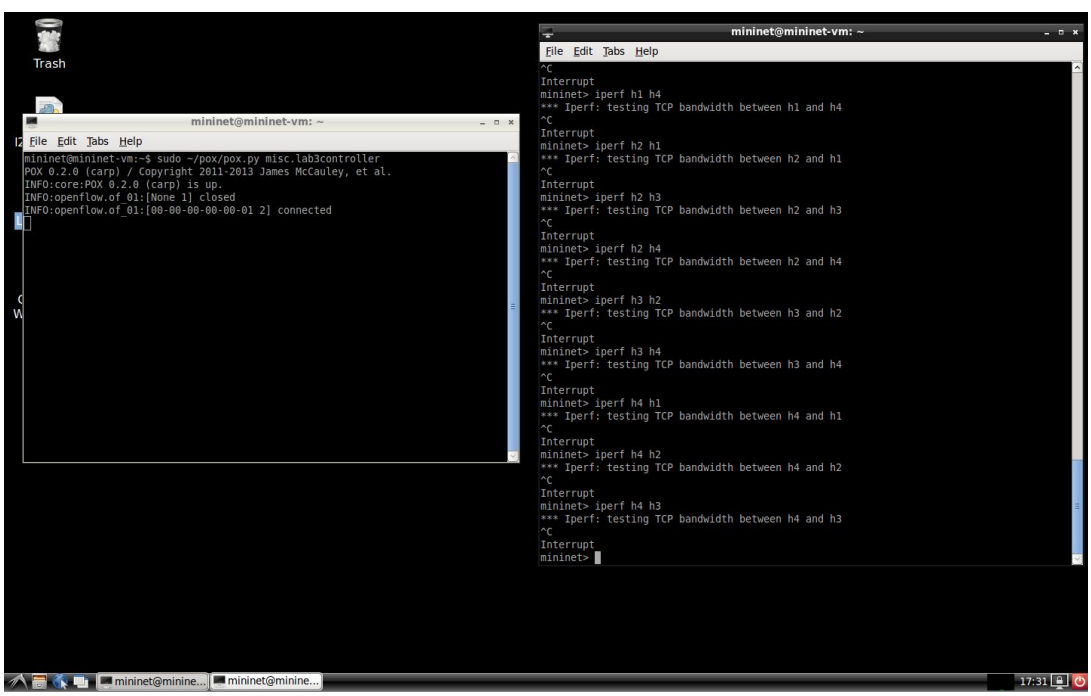

Trash
mininet@mininet-vm: ~
File Edit Tabs Help
mininet@mininet-vm:~$ sudo ~/pox/pox.py misc.lab3controller
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:core:POX 0.2.0 (carp) is up.
INFO:openflow.of_01:[None 1] closed
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
mininet@mininet-vm: ~
File Edit Tabs Help
cookie=0x0, duration=4.888s, table=0, n_packets=1, n_bytes=98, idle_timeout=25,
hard_timeout=50, idle_age=4, icmp,vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_d
st=00:00:00:00:00:04,nw_src=10.0.1.20,nw_dst=10.0.1.40,nw_tos=0,icmp_type=0,icmp
_code=0 actions=FLOOD
cookie=0x0, duration=4.936s, table=0, n_packets=1, n_bytes=98, idle_timeout=25,
hard_timeout=50, idle_age=4, icmp,vlan_tci=0x0000,dl_src=00:00:00:00:00:03,dl_d
st=00:00:00:00:00:02,nw_src=10.0.1.30,nw_dst=10.0.1.20,nw_tos=0,icmp_type=8,icmp
_code=0 actions=FLOOD
cookie=0x0, duration=4.877s, table=0, n_packets=1, n_bytes=98, idle_timeout=25,
hard_timeout=50, idle_age=4, icmp,vlan_tci=0x0000,dl_src=00:00:00:00:00:03,dl_d
st=00:00:00:00:00:04,nw_src=10.0.1.30,nw_dst=10.0.1.40,nw_tos=0,icmp_type=0,icmp
_code=0 actions=FLOOD
cookie=0x0, duration=4.895s, table=0, n_packets=1, n_bytes=98, idle_timeout=25,
hard_timeout=50, idle_age=4, icmp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_d
st=00:00:00:00:00:04,nw_src=10.0.1.10,nw_dst=10.0.1.40,nw_tos=0,icmp_type=0,icmp
_code=0 actions=FLOOD
cookie=0x0, duration=5.093s, table=0, n_packets=1, n_bytes=98, idle_timeout=25,
hard_timeout=50, idle_age=5, icmp,vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_d
st=00:00:00:00:00:01,nw_src=10.0.1.20,nw_dst=10.0.1.10,nw_tos=0,icmp_type=0,icmp
_code=0 actions=FLOOD
cookie=0x0, duration=4.995s, table=0, n_packets=1, n_bytes=98, idle_timeout=25,
hard_timeout=50, idle_age=4, icmp,vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_d
st=00:00:00:00:00:01,nw_src=10.0.1.20,nw_dst=10.0.1.10,nw_tos=0,icmp_type=8,icmp
_code=0 actions=FLOOD
cookie=0x0, duration=4.945s, table=0, n_packets=1, n_bytes=98, idle_timeout=25,
hard_timeout=50, idle_age=4, icmp,vlan_tci=0x0000,dl_src=00:00:00:00:00:03,dl_d
st=00:00:00:00:00:01,nw_src=10.0.1.30,nw_dst=10.0.1.10,nw_tos=0,icmp_type=8,icmp
_code=0 actions=FLOOD
cookie=0x0, duration=4.898s, table=0, n_packets=1, n_bytes=98, idle_timeout=25,
hard_timeout=50, idle_age=4, icmp,vlan_tci=0x0000,dl_src=00:00:00:00:00:04,dl_d
st=00:00:00:00:00:01,nw_src=10.0.1.40,nw_dst=10.0.1.10,nw_tos=0,icmp_type=8,icmp
_code=0 actions=FLOOD
cookie=0x0, duration=5.016s, table=0, n_packets=1, n_bytes=98, idle_timeout=25,
hard_timeout=50, idle_age=5, icmp,vlan_tci=0x0000,dl_src=00:00:00:00:00:03,dl_d
st=00:00:00:00:00:01,nw_src=10.0.1.30,nw_dst=10.0.1.10,nw_tos=0,icmp_type=0,icmp
_code=0 actions=FLOOD
cookie=0x0, duration=5.08s, table=0, n_packets=1, n_bytes=42, idle_timeout=25,
hard_timeout=50, idle_age=5, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:03,dl_dst
=00:00:00:00:00:01,arp_spa=10.0.1.30,arp_tpa=10.0.1.10,arp_op=2 actions=FLOOD
mininet@minine...
mininet@minine...
17:24


Trash
mininet@mininet-vm: ~
File Edit Tabs Help
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
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
mininet@mininet-vm: ~
File Edit Tabs Help
st=00:00:00:00:00:01,nw_src=10.0.1.30,nw_dst=10.0.1.10,nw_tos=0,icmp_type=0,icmp
_code=0 actions=FLOOD
cookie=0x0, duration=5.08s, table=0, n_packets=1, n_bytes=42, idle_timeout=25,
hard_timeout=50, idle_age=5, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:03,dl_dst
=00:00:00:00:00:01,arp_spa=10.0.1.30,arp_tpa=10.0.1.10,arp_op=2 actions=FLOOD
cookie=0x0, duration=5.003s, table=0, n_packets=1, n_bytes=42, idle_timeout=25,
hard_timeout=50, idle_age=5, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:04,dl_ds
t=00:00:00:00:00:01,arp_spa=10.0.1.40,arp_tpa=10.0.1.10,arp_op=2 actions=FLOOD
cookie=0x0, duration=4.987s, table=0, n_packets=1, n_bytes=42, idle_timeout=25,
hard_timeout=50, idle_age=4, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_ds
t=ff:ff:ff:ff:ff:ff,arp_spa=10.0.1.20,arp_tpa=10.0.1.30,arp_op=1 actions=FLOOD
cookie=0x0, duration=5.006s, table=0, n_packets=1, n_bytes=42, idle_timeout=25,
hard_timeout=50, idle_age=5, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_ds
t=ff:ff:ff:ff:ff:ff,arp_spa=10.0.1.10,arp_tpa=10.0.1.40,arp_op=1 actions=FLOOD
cookie=0x0, duration=4.963s, table=0, n_packets=1, n_bytes=42, idle_timeout=25,
hard_timeout=50, idle_age=4, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:04,dl_ds
t=00:00:00:00:00:02,arp_spa=10.0.1.40,arp_tpa=10.0.1.20,arp_op=2 actions=FLOOD
cookie=0x0, duration=5.1s, table=0, n_packets=1, n_bytes=42, idle_timeout=25, h
ard_timeout=50, idle_age=5, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=
00:00:00:00:00:01,arp_spa=10.0.1.20,arp_tpa=10.0.1.10,arp_op=2 actions=FLOOD
cookie=0x0, duration=5.103s, table=0, n_packets=1, n_bytes=42, idle_timeout=25,
hard_timeout=50, idle_age=5, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_ds
t=ff:ff:ff:ff:ff:ff,arp_spa=10.0.1.10,arp_tpa=10.0.1.20,arp_op=1 actions=FLOOD
cookie=0x0, duration=5.083s, table=0, n_packets=1, n_bytes=42, idle_timeout=25,
hard_timeout=50, idle_age=5, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:01,dl_ds
t=ff:ff:ff:ff:ff:ff,arp_spa=10.0.1.10,arp_tpa=10.0.1.30,arp_op=1 actions=FLOOD
cookie=0x0, duration=4.984s, table=0, n_packets=1, n_bytes=42, idle_timeout=25,
hard_timeout=50, idle_age=4, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:03,dl_ds
t=00:00:00:00:00:02,arp_spa=10.0.1.30,arp_tpa=10.0.1.20,arp_op=2 actions=FLOOD
cookie=0x0, duration=4.912s, table=0, n_packets=1, n_bytes=42, idle_timeout=25,
hard_timeout=50, idle_age=4, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:04,dl_ds
t=00:00:00:00:00:03,arp_spa=10.0.1.40,arp_tpa=10.0.1.30,arp_op=2 actions=FLOOD
cookie=0x0, duration=4.965s, table=0, n_packets=1, n_bytes=42, idle_timeout=25,
hard_timeout=50, idle_age=4, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:02,dl_ds
t=ff:ff:ff:ff:ff:ff,arp_spa=10.0.1.20,arp_tpa=10.0.1.40,arp_op=1 actions=FLOOD
cookie=0x0, duration=4.915s, table=0, n_packets=1, n_bytes=42, idle_timeout=25,
hard_timeout=50, idle_age=4, arp,vlan_tci=0x0000,dl_src=00:00:00:00:00:03,dl_ds
t=ff:ff:ff:ff:ff:ff,arp_spa=10.0.1.30,arp_tpa=10.0.1.40,arp_op=1 actions=FLOOD
mininet>
mininet@minine...
mininet@minine...
17:24

Iperf:

This command is used to test the TCP connection between different hosts. When I tested the TCP connections between host 1 and host 3, the TCP connection works. However, when testing it with any other hosts, we see that the connection will hang for an indefinite amount of time, meaning that the packet was dropped.





Screenshots of my Code:

lab3controller.py

File Edit Search Options Help

```python
# Lab 3 Skeleton
#
# Based on of_tutorial by James McCauley

##########################################################################################
# Shawn Chumbar
# CMPE 150: Lab 3
# Professor Christina Parsa
# lab3controller.py
##########################################################################################

from pox.core import core
import pox.openflow.libopenflow_01 as of

log = core.getLogger()


class Firewall (object):
  """
  A Firewall object is created for each switch that connects.
  A Connection object for that switch is passed to the __init__ function.
  """

  def __init__(self, connection):
        # Keep track of the connection to the switch so that we can
        # send it messages
        self.connection = connection

        # This binds our PacketIn event listener
        connection.addListeners(self)

  def do_firewall(self, packet, packet_in):
        ##########################################################################################
        #                                                Rules:
        #1. Allow any ARP traffic.
        #2. Allow any ICMP traffic.
        #3. TCP traffic should be allowed between h1 and h3 (both h1 to h3 and h3 to h1).
        #4. Any other traffic should be dropped irrespective of the protocol.
        ##########################################################################################
        # The code in here will be executed for every packet.

        # Making the table
```

misc lab3controller.py 17:45

lab3controller.py

File Edit Search Options Help

```python
  def do_firewall(self, packet, packet_in):
        ##########################################################################################
        #                                                Rules:
        #1. Allow any ARP traffic.
        #2. Allow any ICMP traffic.
        #3. TCP traffic should be allowed between h1 and h3 (both h1 to h3 and h3 to h1).
        #4. Any other traffic should be dropped irrespective of the protocol.
        ##########################################################################################
        # The code in here will be executed for every packet.

        # Making the table
        # Installing table entry

        msg = of.ofp_flow_mod()

        # match packet
        msg.match = of.ofp_match.from_packet(packet)

        # few entries of dump--flows from timeout
        msg.idle_timeout = 25  # 50
        msg.hard_timeout = 50  # 100

        isARP = packet.find('arp')
        isTCP = packet.find('tcp')
        isICMP = packet.find('icmp')

        # If we are using ARP, TCP, or ICMP
        if isICMP is not None or isARP is not None or isTCP is not None:
                # 2.Allow any ICMP traffic
                if isICMP is not None:
                # accept packet
                        msg.data = packet_in
                        msg.nw_proto = 6 #6=tcp
                        # add action to send to specified port
                        action = of.ofp_action_output(port = of.OFPP_FLOOD)
                        msg.actions.append(action)
                        self.connection.send(msg)

                #1. Allow any ARP traffic
                elif isARP is not None:
                # accept packet
                        msg.data = packet_in
```

misc lab3controller.py 17:46

lab3controller.py

File Edit Search Options Help

```
                #1. Allow any ARP traffic
                elif isARP is not None:
                # accept packet
                        msg.data = packet_in
                        msg.match.dl_type = 0x0806  # match ARP
                        # add action to send to specified port
                        action = of.ofp_action_output(port=of.OFPP_FLOOD)
                        msg.actions.append(action)
                        self.connection.send(msg)

                # 3. TCP traffic should be allowed between h1 and h3 (both h1 to h3 and h3 to h1).
                elif isTCP is not None:
                # accept packet
                        isIPV4 = packet.find('ipv4')
                        if (isIPV4.srcip == '10.0.1.10' and isIPV4.dstip == '10.0.1.30') or (isIPV4.srcip == '10.0.1.30' and isIPV4.dstip
                                msg.data = packet_in
                                msg.nw_proto = 6 # 6 = tcp
                                # add action to send to specified port
                                action = of.ofp_action_output(port = of.OFPP_FLOOD)
                                msg.actions.append(action)
                                self.connection.send(msg)
                else:
                        # no packets taken - packet dropped
                        # msg.data = packet_in
                        print("\n\n\nDROP TCP\n\n\n")
                        self.connection.send(msg)

        #4. Any other traffic should be dropped irrespective of the protocol.
        else:
        #       # msg.data = packet_in
                self.connection.send(msg)

        ##################################################

  def _handle_PacketIn (self, event):
        """
        Handles packet in messages from the switch.
        """

        packet = event.parsed # This is the parsed packet data.
        if not packet.parsed:
          log.warning("Ignoring incomplete packet")
```

misc | lab3controller.py | 17:46

lab3controller.py

File Edit Search Options Help

```
                        if (isIPV4.srcip == '10.0.1.10' and isIPV4.dstip == '10.0.1.30') or (isIPV4.srcip == '10.0.1.30' and isIPV4.dstip
                                msg.data = packet_in
                                msg.nw_proto = 6 # 6 = tcp
                                # add action to send to specified port
                                action = of.ofp_action_output(port = of.OFPP_FLOOD)
                                msg.actions.append(action)
                                self.connection.send(msg)
                else:
                        # no packets taken - packet dropped
                        # msg.data = packet_in
                        print("\n\n\nDROP TCP\n\n\n")
                        self.connection.send(msg)

        #4. Any other traffic should be dropped irrespective of the protocol.
        else:
        #       # msg.data = packet_in
                self.connection.send(msg)

        ##################################################

  def _handle_PacketIn (self, event):
        """
        Handles packet in messages from the switch.
        """

        packet = event.parsed # This is the parsed packet data.
        if not packet.parsed:
          log.warning("Ignoring incomplete packet")
          return

        packet_in = event.ofp # The actual ofp_packet_in message.
        self.do_firewall(packet, packet_in)

def launch ():
  """
  Starts the component
  """
  def start_switch (event):
        log.debug("Controlling %s" % (event.connection,))
        Firewall(event.connection)
  core.openflow.addListenerByName("ConnectionUp", start_switch)
```

misc | lab3controller.py | 17:46