Alex Davila
1465932
CSE150/L
Chen Qian

Introduction to Computer Networks Lab

Lab 3 - Simple Firewall using OpenFlow

For this assignment I created a simple firewall using OpenFlow-enabled switches. The term "firewall" is derived from building construction: a firewall is a wall you place in buildings to stop a fire from spreading. In the case of networking, it is the act of providing security by not letting specified traffic pass through the firewall. This feature is good for minimizing attack vectors and limiting the network "surface" exposed to attackers.

In this Lab, I was provided with the Mininet configuration, lab3.py, to setup the network which assumes a remote controller listening on the default IP address and port number 127.0.0.1:6633.

In this Lab, I was also provided with a skeleton POX controller: lab3controller.py. This file was where I created my modifications to the firewall.

Running the Code:

1. To run the controller, I placed lab3controller.py in the ~/pox/pox/misc directory. I could then launch the controller with the command **sudo ~/pox/pox.py misc.lab3controller**
2. To run the mininet file, I placed it in ~ and ran the command **sudo python ~/lab3.py**

To do this assignment, I ran both files at the same time (in 2 different terminal windows).

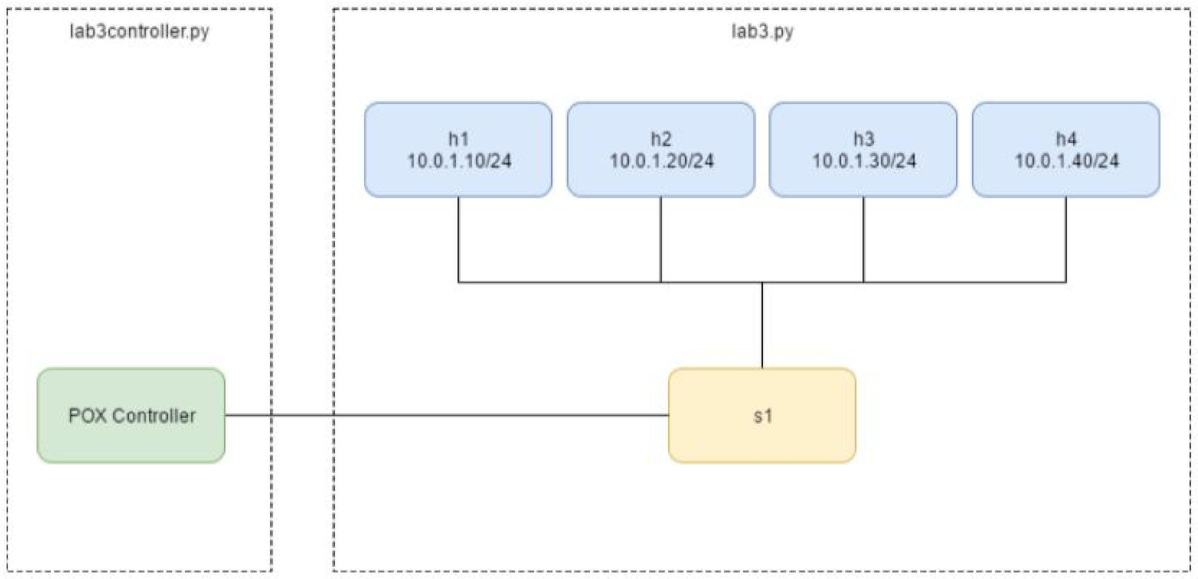
Basically, the Firewall allows all ARP and TCP traffic to pass. However, any other type of traffic is dropped. It as acceptable to flood the allowable traffic out all ports.

Flow tables match the rule with the highest priority first, where priority is established based on the order rules are placed in the table.

When I created a rule in the POX controller, I also had POX "install" the rule in the switch. This makes it so the switch "remembers" what to do for a few seconds. The switch simply does not always ask the controller what to do for every packet it receives.

To do this, I used ofp_flow_mod.

The topology that will be created will look as follows:



The Rules:

The rules that I implemented in OpenFlow for this assignment are:

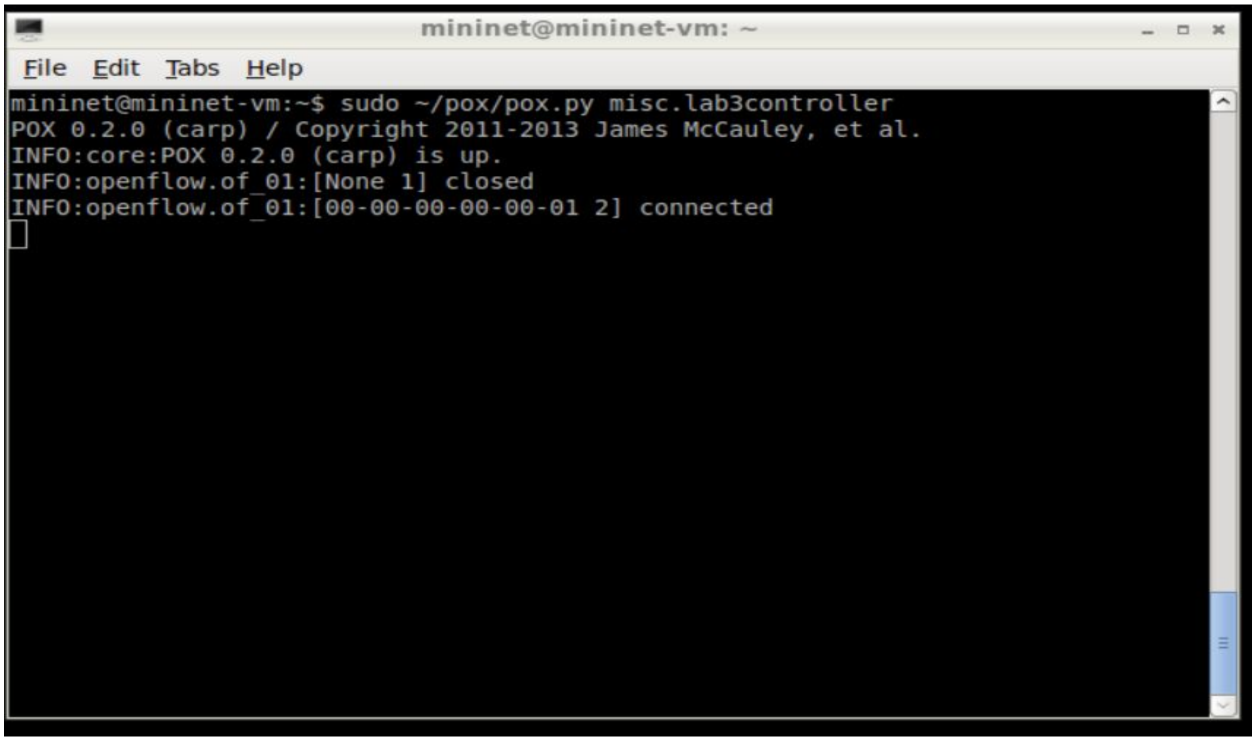
src ip	**dst ip**	**protocol**	**action**
ipv4	ipv4	tcp	accept
any	any	arp	accept
any ipv4	any ipv4	-	drop

Useful Resources:

http://intronetworks.cs.luc.edu/auxiliary_files/mininet/poxwiki.pdf
Inside VM, the pox/forwarding/l2_learning.py example file.

Testing/Submission/Grading

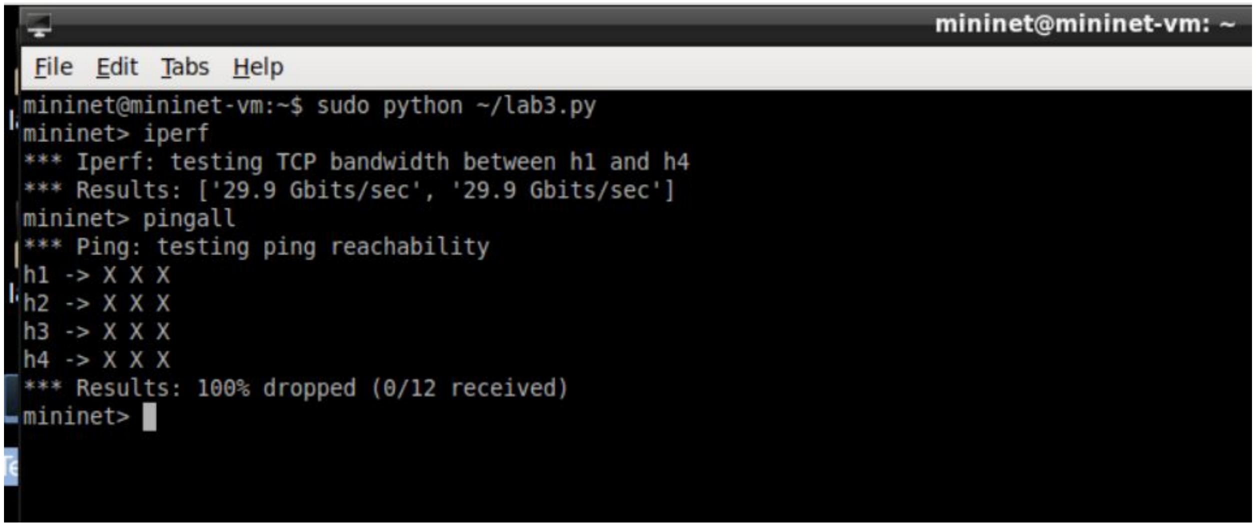
To test your controller, first start the controller, then start the mininet script. When you are prompted with the mininet CLI, run the following commands and take a screenshot of each:



```
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
```

[30 points] pingall : This should fail, since ICMP traffic should be blocked.

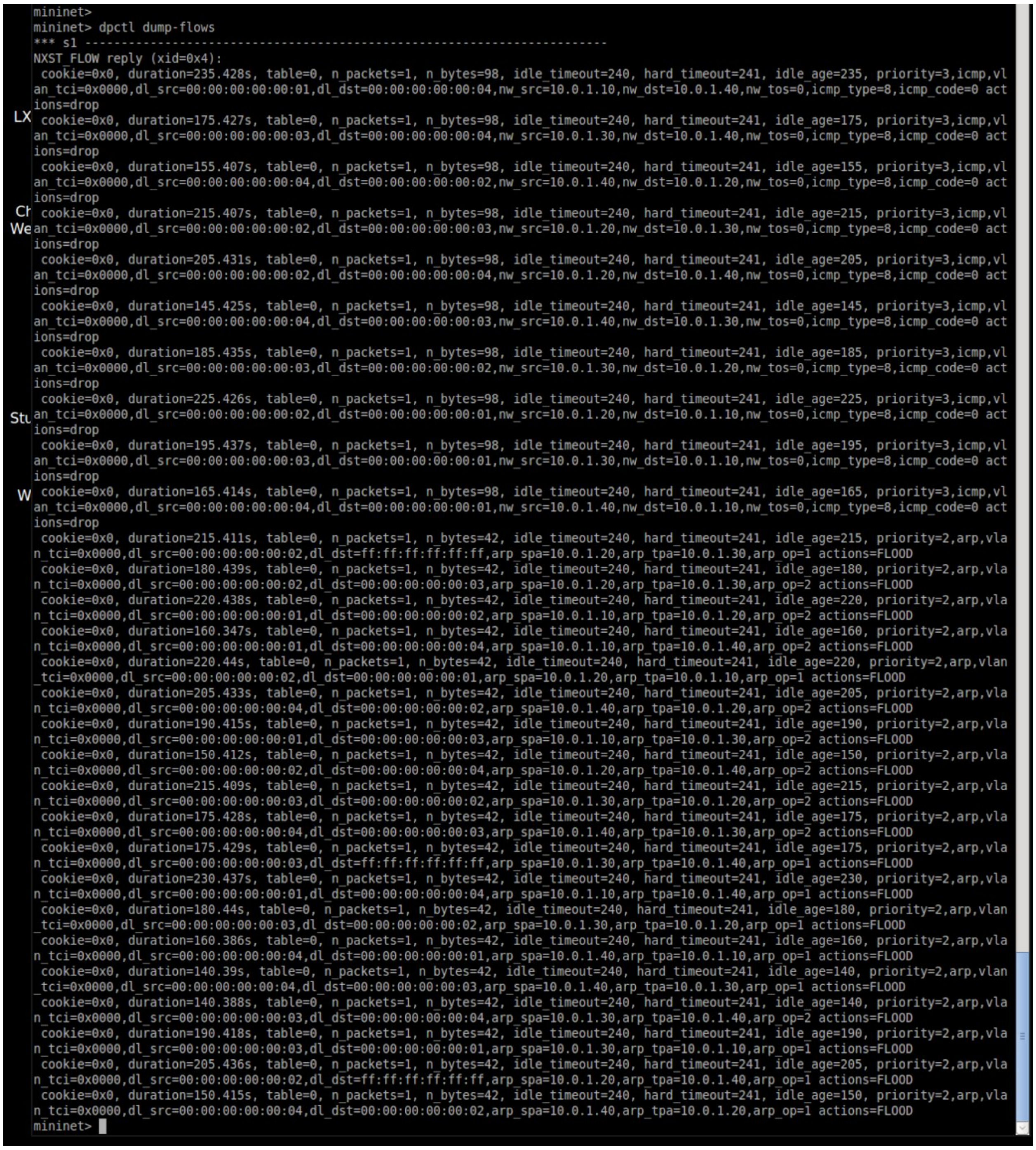
- 20 points: ping succeeds
- The remaining 10 points will be awarded depending on the quality of the explanation given.



```
mininet@mininet-vm:~$ sudo python ~/lab3.py
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['29.9 Gbits/sec', '29.9 Gbits/sec']
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)
mininet>
```

[70 points] dpctl dump--flows : This should show a few entries. These are the entries that you installed into the switch with of_flow_mod. You'll need to do this within the timeout you specified in your of_flow_mod for the entries to show up!

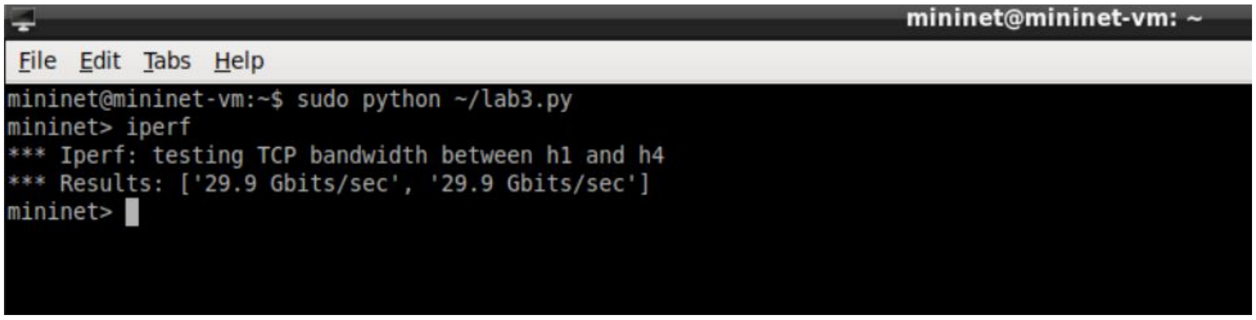
- 40 points: no flows shown
- The remaining 30 points will be awarded depending on the quality of the explanation given.



```
mininet>
mininet> dpctl dump-flows
*** s1 ------------------------------------------------------------------------
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=235.428s, table=0, n_packets=1, n_bytes=98, idle_timeout=240, hard_timeout=241, idle_age=235, priority=3,icmp,vl
an_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:04,nw_src=10.0.1.10,nw_dst=10.0.1.40,nw_tos=0,icmp_type=8,icmp_code=0 act
ions=drop
 cookie=0x0, duration=175.427s, table=0, n_packets=1, n_bytes=98, idle_timeout=240, hard_timeout=241, idle_age=175, priority=3,icmp,vl
an_tci=0x0000,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:04,nw_src=10.0.1.30,nw_dst=10.0.1.40,nw_tos=0,icmp_type=8,icmp_code=0 act
ions=drop
 cookie=0x0, duration=155.407s, table=0, n_packets=1, n_bytes=98, idle_timeout=240, hard_timeout=241, idle_age=155, priority=3,icmp,vl
an_tci=0x0000,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:02,nw_src=10.0.1.40,nw_dst=10.0.1.20,nw_tos=0,icmp_type=8,icmp_code=0 act
ions=drop
 cookie=0x0, duration=215.407s, table=0, n_packets=1, n_bytes=98, idle_timeout=240, hard_timeout=241, idle_age=215, priority=3,icmp,vl
an_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03,nw_src=10.0.1.20,nw_dst=10.0.1.30,nw_tos=0,icmp_type=8,icmp_code=0 act
ions=drop
 cookie=0x0, duration=205.431s, table=0, n_packets=1, n_bytes=98, idle_timeout=240, hard_timeout=241, idle_age=205, priority=3,icmp,vl
an_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:04,nw_src=10.0.1.20,nw_dst=10.0.1.40,nw_tos=0,icmp_type=8,icmp_code=0 act
ions=drop
 cookie=0x0, duration=145.425s, table=0, n_packets=1, n_bytes=98, idle_timeout=240, hard_timeout=241, idle_age=145, priority=3,icmp,vl
an_tci=0x0000,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:03,nw_src=10.0.1.40,nw_dst=10.0.1.30,nw_tos=0,icmp_type=8,icmp_code=0 act
ions=drop
 cookie=0x0, duration=185.435s, table=0, n_packets=1, n_bytes=98, idle_timeout=240, hard_timeout=241, idle_age=185, priority=3,icmp,vl
an_tci=0x0000,dl_src=00:00:00:00:00:03,dl_dst=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 act
ions=drop
 cookie=0x0, duration=225.426s, table=0, n_packets=1, n_bytes=98, idle_timeout=240, hard_timeout=241, idle_age=225, priority=3,icmp,vl
an_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=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 act
ions=drop
 cookie=0x0, duration=195.437s, table=0, n_packets=1, n_bytes=98, idle_timeout=240, hard_timeout=241, idle_age=195, priority=3,icmp,vl
an_tci=0x0000,dl_src=00:00:00:00:00:03,dl_dst=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 act
ions=drop
 cookie=0x0, duration=165.414s, table=0, n_packets=1, n_bytes=98, idle_timeout=240, hard_timeout=241, idle_age=165, priority=3,icmp,vl
an_tci=0x0000,dl_src=00:00:00:00:00:04,dl_dst=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 act
ions=drop
 cookie=0x0, duration=215.411s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=215, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=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=180.439s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=180, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:03,arp_spa=10.0.1.20,arp_tpa=10.0.1.30,arp_op=2 actions=FLOOD
 cookie=0x0, duration=220.438s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=220, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02,arp_spa=10.0.1.10,arp_tpa=10.0.1.20,arp_op=2 actions=FLOOD
 cookie=0x0, duration=160.347s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=160, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:04,arp_spa=10.0.1.10,arp_tpa=10.0.1.40,arp_op=2 actions=FLOOD
 cookie=0x0, duration=220.44s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=220, priority=2,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=1 actions=FLOOD
 cookie=0x0, duration=205.433s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=205, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:04,dl_dst=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=190.415s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=190, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:03,arp_spa=10.0.1.10,arp_tpa=10.0.1.30,arp_op=2 actions=FLOOD
 cookie=0x0, duration=150.412s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=150, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:04,arp_spa=10.0.1.20,arp_tpa=10.0.1.40,arp_op=2 actions=FLOOD
 cookie=0x0, duration=215.409s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=215, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:03,dl_dst=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=175.428s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=175, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:04,dl_dst=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=175.429s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=175, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:03,dl_dst=ff:ff:ff:ff:ff:ff,arp_spa=10.0.1.30,arp_tpa=10.0.1.40,arp_op=1 actions=FLOOD
 cookie=0x0, duration=230.437s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=230, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:04,arp_spa=10.0.1.10,arp_tpa=10.0.1.40,arp_op=1 actions=FLOOD
 cookie=0x0, duration=180.44s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=180, priority=2,arp,vlan
_tci=0x0000,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:02,arp_spa=10.0.1.30,arp_tpa=10.0.1.20,arp_op=1 actions=FLOOD
 cookie=0x0, duration=160.386s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=160, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:01,arp_spa=10.0.1.40,arp_tpa=10.0.1.10,arp_op=1 actions=FLOOD
 cookie=0x0, duration=140.39s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=140, priority=2,arp,vlan
_tci=0x0000,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:03,arp_spa=10.0.1.40,arp_tpa=10.0.1.30,arp_op=1 actions=FLOOD
 cookie=0x0, duration=140.388s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=140, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:03,dl_dst=00:00:00:00:00:04,arp_spa=10.0.1.30,arp_tpa=10.0.1.40,arp_op=2 actions=FLOOD
 cookie=0x0, duration=190.418s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=190, priority=2,arp,vla
n_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=1 actions=FLOOD
 cookie=0x0, duration=205.436s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=205, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:02,dl_dst=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=150.415s, table=0, n_packets=1, n_bytes=42, idle_timeout=240, hard_timeout=241, idle_age=150, priority=2,arp,vla
n_tci=0x0000,dl_src=00:00:00:00:00:04,dl_dst=00:00:00:00:00:02,arp_spa=10.0.1.40,arp_tpa=10.0.1.20,arp_op=1 actions=FLOOD
mininet> 
```

[70 points] iperf : This should succeed.

- 40 points: iperf fails
- The remaining 30 points will be awarded depending on the quality of the explanation given.

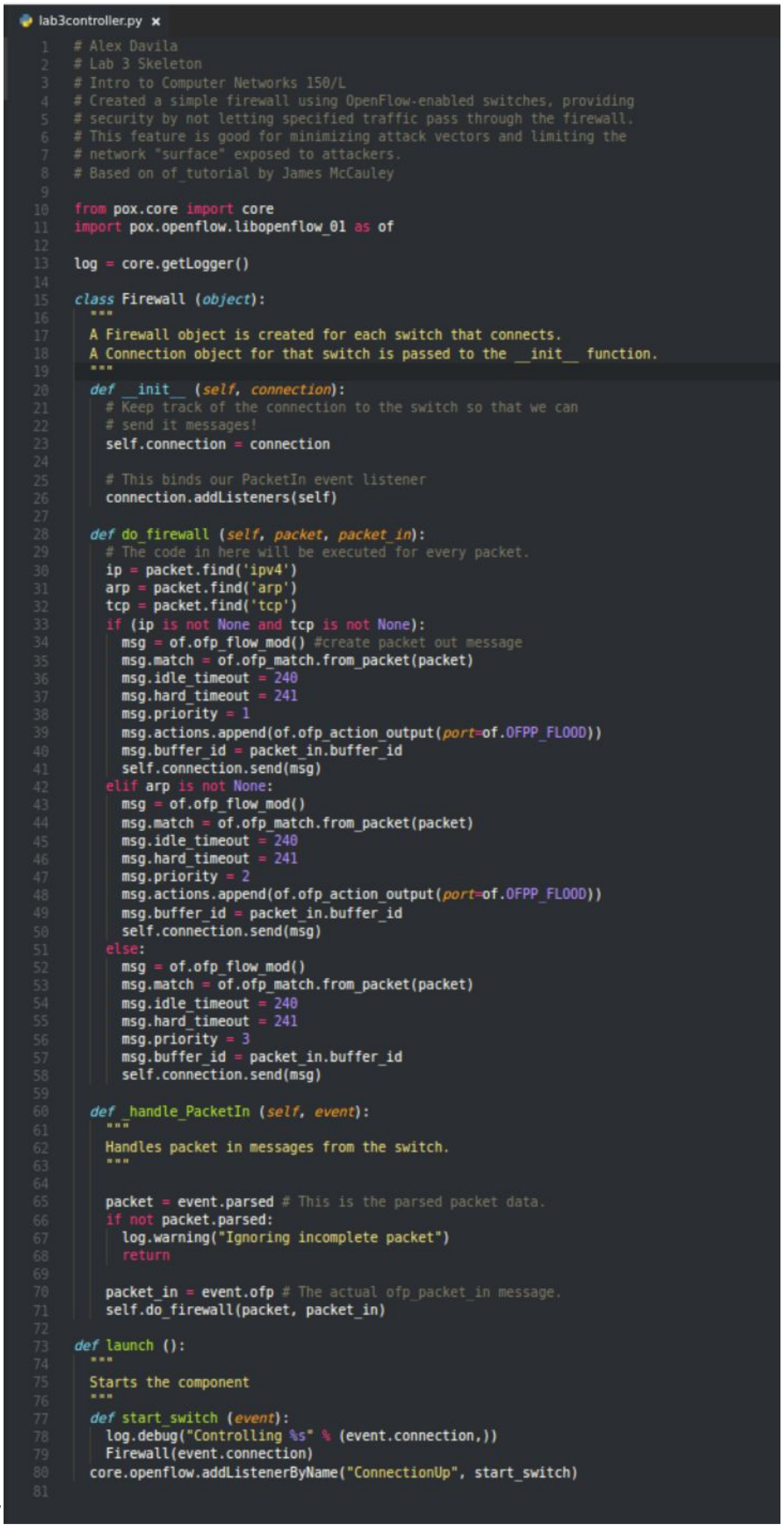


```
mininet@mininet-vm:~$ sudo python ~/lab3.py
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['29.9 Gbits/sec', '29.9 Gbits/sec']
mininet> 
```

Additionally, you must submit your firewall code. It should be named **lab3controller.py.**
Note: Your code will be tested. Ensure the screenshots are of your own code. Submitting screenshots that are not of your own code is considered cheating and a violation of the academic integrity agreement.

[30 points] Firewall code and screenshots submitted and named properly:

- 10 points: lab3.pdf wrong format or name.
- 10 points: lab3controller.py wrong format, wrong name, or missing.
- 10 points: README not submitted


lab3controller.py

```python
# Alex Davila
# Lab 3 Skeleton
# Intro to Computer Networks 150/L
# Created a simple firewall using OpenFlow-enabled switches, providing
# security by not letting specified traffic pass through the firewall.
# This feature is good for minimizing attack vectors and limiting the
# network "surface" exposed to attackers.
# Based on of_tutorial by James McCauley

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):
    # The code in here will be executed for every packet.
    ip = packet.find('ipv4')
    arp = packet.find('arp')
    tcp = packet.find('tcp')
    if (ip is not None and tcp is not None):
      msg = of.ofp_flow_mod() #create packet out message
      msg.match = of.ofp_match.from_packet(packet)
      msg.idle_timeout = 240
      msg.hard_timeout = 241
      msg.priority = 1
      msg.actions.append(of.ofp_action_output(port=of.OFPP_FLOOD))
      msg.buffer_id = packet_in.buffer_id
      self.connection.send(msg)
    elif arp is not None:
      msg = of.ofp_flow_mod()
      msg.match = of.ofp_match.from_packet(packet)
      msg.idle_timeout = 240
      msg.hard_timeout = 241
      msg.priority = 2
      msg.actions.append(of.ofp_action_output(port=of.OFPP_FLOOD))
      msg.buffer_id = packet_in.buffer_id
      self.connection.send(msg)
    else:
      msg = of.ofp_flow_mod()
      msg.match = of.ofp_match.from_packet(packet)
      msg.idle_timeout = 240
      msg.hard_timeout = 241
      msg.priority = 3
      msg.buffer_id = packet_in.buffer_id
      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)

```

[lab3controller.py]

Deliverables:

1. **lab3.pdf**: Screenshots of the above commands. If you are not able to get the expected results, below your screenshot, explain what you think is going on (for partial credit).

2. **lab3controller.py:** Your firewall code.

3. **README (or README.txt)**: A readme file explaining your submission.

Note: You do not need to submit lab3.py. You SHOULD NOT modify lab3.py.