

# CSCI558L Lab9

11/06/2011

ESHA DESAI  
SUELA BUZI  
LEON ABURIME  
HARSH GUPTA

**Contents**

Objective.....3

Code.....3

Observations.....6

References.....8

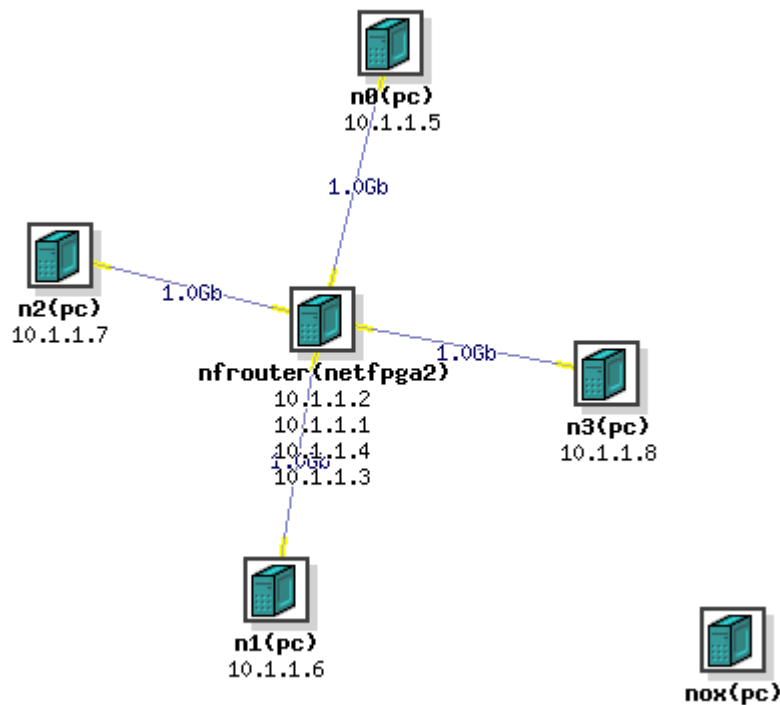
Conclusion.....8

## Objective:

1. Creating a learning Switch with Open Flow Switch on NetFPGA.
2. Creating a Firewall on the learning Open Flow Switch

## Visualization:

The following is the snapshot of the visulaiztion of the topology created bu the ns file provided for the OpenFlow experiment.



Nox is an open source Open Flow controller intended to simplify the creation of software for controlling or monitoring networks. Programs written with NOX have flow-level control of the network. This means that they can determine which flows are allowed on the network.

## Code:

```
# Tutorial Controller
# Starts as a hub, and your job is to turn this into a learning switch.

import logging

from nox.lib.core import *
import nox.lib.openflow as openflow
from nox.lib.packet.ethernet import ethernet
from nox.lib.packet.packet_utils import mac_to_str, mac_to_int
```

```

from twisted.python import log

import logging
from time import time
from socket import htons
from struct import unpack

log = logging.getLogger('nox.coreapps.tutorial.pytutorial')
CACHE_TIMEOUT = 5

class pytutorial(Component):

    def __init__(self, ctxt):
        global inst
        Component.__init__(self, ctxt)

        # Use this table to store MAC addresses in the format of your choice;
        # Functions already imported, including mac_to_str, and mac_to_int,
        # should prove useful for converting the byte array provided by NOX
        # for packet MAC destination fields.
        # This table is initialized to empty when your module starts up.
        self.mac_to_port = {} # key: MAC addr; value: port
    def learn_and_forward(self, dpid, inport, packet, buf, bufid):
        """Learn MAC src port mapping, then flood or send unicast."""
        global inst
        # Initial hub behavior: flood packet out everything but input port.
        # Comment out the line below when starting the exercise.
        # self.send_openflow(dpid, bufid, buf, openflow.OFPP_FLOOD, inport)

        # Starter psuedocode for learning switch exercise below: you'll need
to
        # replace each pseudocode line with more specific Python code.

        # Learn the port for the source MAC
        srcaddr = packet.src.tostring()
        if ord(srcaddr[0]) & 1:
            return
        #self.mac_to_port = <fill in>
        if not self.mac_to_port.has_key(srcaddr):
            self.mac_to_port[srcaddr] = inport
        #forward
        dstaddr = packet.dst.tostring()
        #if (destination MAC of the packet is known):
        if not ord(dstaddr[0]) & 1 and self.mac_to_port.has_key(dstaddr):
            #Send unicast packet to known output port
            #self.send_openflow( <fill in params> )
            prt = self.mac_to_port[dstaddr]
            flow = extract_flow(packet)
            flow[core.IN_PORT] = inport
            actions = [[openflow.OFPAT_OUTPUT, [0, prt]]]
            # BLOCK UDP and 9999
            if (flow[core.NW_PROTO]==ipv4.ipv4.UDP_PROTOCOL and
flow[core.TP_DST]==9999):
                actions=[]

```

```

        # Later, only after learning controller works:
        # push down flow entry and remove the send_openflow command
above.

        #self.install_datapath_flow ( <fill in params> )
        self.install_datapath_flow(dpid, flow, CACHE_TIMEOUT,
                                   openflow.OFP_FLOW_PERMANENT, actions,
                                   bufid, openflow.OFP_DEFAULT_PRIORITY,
                                   inport, buf)
    else:
        #flood packet out everything but the input port
        self.send_openflow(dpid, bufid, buf, openflow.OFPP_FLOOD, inport)

def packet_in_callback(self, dpid, inport, reason, len, bufid, packet):
    """Packet-in handler"""
    if not packet.parsed:
        log.debug('Ignoring incomplete packet')
    else:
        self.learn_and_forward(dpid, inport, packet, packet.arr, bufid)

    return CONTINUE

def install(self):
    self.register_for_packet_in(self.packet_in_callback)

def getInterface(self):
    return str(pytutorial)

def getFactory():
    class Factory:
        def instance(self, ctxt):
            return pytutorial(ctxt)
    return Factory()

```

## Observations:

### 1. Creating a learning Switch with Open Flow Switch on NetFPGA.

Before the modifications to the code in pyswitch.py, the switch behaved like a hub. We saw that when we tried pinging from node0 to node2, the packet was being received by not only node2 but by all the other nodes as well. All nodes did receive the ICMP packet but only node2 replied back.

Following is the snapshot of the **tcpdump** results at the interfaces of the switch facing node2 and node3 (Interface names : **nf2c2** and **nf2c3**) :

```
sc558ag@n0:~$ ping -i 10 -c 7 n2
PING n2-lan2 (10.1.1.7) 56(84) bytes of data.
64 bytes from n2-lan2 (10.1.1.7): icmp_seq=1 ttl=64 time=2.88 ms
64 bytes from n2-lan2 (10.1.1.7): icmp_seq=2 ttl=64 time=3.05 ms
^C
--- n2-lan2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 10010ms
rtt min/avg/max/mdev = 2.882/2.969/3.057/0.103 ms
sc558ag@n0:~$
```

```
-bash-3.2$ sudo tcpdump -vv -i nf2c2
tcpdump: WARNING: nf2c2: no IPv4 address assigned
tcpdump: listening on nf2c2, link-type EN10MB (Ethernet), capture size 96 bytes
23:11:50.915695 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto: ICMP (1), length: 84) n0-lan0 > n2-lan2: ICMP echo request, id 63050, seq 1, length 64
23:11:50.915977 IP (tos 0x0, ttl 64, id 9820, offset 0, flags [none], proto: ICMP (1), length: 84) n2-lan2 > n0-lan0: ICMP echo reply, id 63050, seq 1, length 64
23:11:55.913071 arp who-has n2-lan2 tell n0-lan0
23:11:55.913175 arp reply n2-lan2 is-at 00:15:17:57:c6:7a (oui Unknown)
23:11:58.704391 CDPv2, ttl: 180s, checksum: 692 (unverified), length 241
Device-ID (0x01), length: 11 bytes: 'SAL0739M2PL'
Address (0x02), length: 13 bytes: IPv4 (1) cisco4.isi.deterlab.net
Port-ID (0x03), length: 3 bytes: '9/3'
Capability (0x04), length: 4 bytes: (0x0000002a): Transparent Bridge, L2
Switch, IGMP snooping[icdp]
23:11:58.706472 CDPv2, ttl: 180s, checksum: 692 (unverified), length 241
Device-ID (0x01), length: 11 bytes: 'SAL0739M2PL'
Address (0x02), length: 13 bytes: IPv4 (1) cisco4.isi.deterlab.net
Port-ID (0x03), length: 3 bytes: '9/3'
Capability (0x04), length: 4 bytes: (0x0000002a): Transparent Bridge, L2
Switch, IGMP snooping[icdp]
```

```
-bash-3.2$ sudo tcpdump -vv -i nf2c3
tcpdump: WARNING: nf2c3: no IPv4 address assigned
tcpdump: listening on nf2c3, link-type EN10MB (Ethernet), capture size 96 bytes
23:11:50.915191 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto: ICMP (1), length: 84) n0-lan0 > n2-lan2: ICMP echo request, id 63050, seq 1, length 64
23:11:58.706503 CDPv2, ttl: 180s, checksum: 692 (unverified), length 241
Device-ID (0x01), length: 11 bytes: 'SAL0739M2PL'
Address (0x02), length: 13 bytes: IPv4 (1) cisco4.isi.deterlab.net
Port-ID (0x03), length: 3 bytes: '9/1'
Capability (0x04), length: 4 bytes: (0x0000002a): Transparent Bridge, L2
Switch, IGMP snooping[icdp]
23:11:58.706875 CDPv2, ttl: 180s, checksum: 692 (unverified), length 241
Device-ID (0x01), length: 11 bytes: 'SAL0739M2PL'
Address (0x02), length: 13 bytes: IPv4 (1) cisco4.isi.deterlab.net
Port-ID (0x03), length: 3 bytes: '9/3'
Capability (0x04), length: 4 bytes: (0x0000002a): Transparent Bridge, L2
Switch, IGMP snooping[icdp]
23:11:59.742055 CDPv2, ttl: 180s, checksum: 692 (unverified), length 241
Device-ID (0x01), length: 11 bytes: 'SAL0739M2PL'
Address (0x02), length: 13 bytes: IPv4 (1) cisco4.isi.deterlab.net
Port-ID (0x03), length: 3 bytes: '9/4'
Capability (0x04), length: 4 bytes: (0x0000002a): Transparent Bridge, L2
Switch, IGMP snooping[icdp]
```

The above snapshot clearly shows the “Flooding” mechanism of the switch acting like a hub.

After our modifications to the code, we observed that the switch started behaving like a “Learning switch”. When we tried pinging from node0 to node3, we observed at the interface of the switch facing node3 (interface name nf2c3), that all the ICMP packets were received and we could see that node3 replied back to node0 too. But when we pinged from node0 to node2, we observed at the interface of the switch facing node3 (interface name nf2c3), that the ICMP request was not received at all. (Please ignore the CDPv2 protocol appearances on the interface)

The image shows two terminal windows. The left window, titled 'usersdeterlab.net - PuTTY', shows the output of a ping command from node0 to node3. The output indicates that 7 packets were transmitted and received with 0% packet loss and an average round-trip time of 2.426ms. Below the ping statistics, there are OpenFlow event logs showing packet-in and flow expired events. The right window, titled 'root@nox:/users/sc558ag', shows the output of a tcpdump command on interface nf2c3. The output shows several ICMP echo requests and replies between node0 and node3, with the switch (node0-lan0) acting as the source or destination for the packets.

```

usersdeterlab.net - PuTTY
sc558ag@n0:~$ ping -i 10 -c 7 n3
PING n3-lan3 (10.1.1.8) 56(84) bytes of data.
64 bytes from n3-lan3 (10.1.1.8): icmp_seq=1 ttl=64 time=1.6 ms
64 bytes from n3-lan3 (10.1.1.8): icmp_seq=2 ttl=64 time=3.04 ms
64 bytes from n3-lan3 (10.1.1.8): icmp_seq=3 ttl=64 time=2.57 ms
64 bytes from n3-lan3 (10.1.1.8): icmp_seq=4 ttl=64 time=3.07 ms
64 bytes from n3-lan3 (10.1.1.8): icmp_seq=5 ttl=64 time=2.78 ms
64 bytes from n3-lan3 (10.1.1.8): icmp_seq=6 ttl=64 time=2.48 ms

--- n2-lan2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 60064ms
rtt min/avg/max/mdev = 2.426/3.590/8.515/2.019 ms

00317/openflow-event[DBG]:received packet-in event from 000000000001 (len:60)
00318/openflow-event[DBG]:received packet-in event from 000000000001 (len:60)
00319/openflow-event[DBG]:received flow expired event from 000000000001
00320/openflow-event[DBG]:received flow expired event from 000000000001

root@nox:/users/sc558ag
-bash-3.2$ sudo tcpdump -vv -i nf2c3
tcpdump: WARNING: nf2c3: no IPv4 address assigned
tcpdump: listening on nf2c3, link-type EN10MB (Ethernet), capture size 96 bytes
22:57:52.921241 arp who-has n3-lan3 tell n0-lan0
22:57:52.923566 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto: ICMP (1), length: 84) n0-lan0 > n3-lan3: ICMP echo request, id 59466, seq 1, length 64
22:57:52.923668 IP (tos 0x0, ttl 64, id 54204, offset 0, flags [none], proto: ICMP (1), length: 84) n3-lan3 > n0-lan0: ICMP echo reply, id 59466, seq 1, length 64
22:57:57.919782 arp who-has n0-lan0 tell n3-lan3
22:57:57.923292 arp reply n0-lan0 is-at 00:15:17:57:c7:56 (oui Unknown)
22:58:02.924043 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto: ICMP (1), length: 84) n0-lan0 > n3-lan3: ICMP echo request, id 59466, seq 2, length 64
22:58:02.924160 IP (tos 0x0, ttl 64, id 54205, offset 0, flags [none], proto: ICMP (1), length: 84) n3-lan3 > n0-lan0: ICMP echo reply, id 59466, seq 2, length 64
22:58:12.932081 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto: ICMP (1), length: 84) n0-lan0 > n3-lan3: ICMP echo request, id 59466, seq 3, length 64
22:58:12.932194 IP (tos 0x0, ttl 64, id 54206, offset 0, flags [none], proto: ICMP (1), length: 84) n3-lan3 > n0-lan0: ICMP echo reply, id 59466, seq 3, length 64
22:58:22.941588 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto: ICMP (1), length: 84) n0-lan0 > n3-lan3: ICMP echo request, id 59466, seq 4, length 64
22:58:22.941721 IP (tos 0x0, ttl 64, id 54207, offset 0, flags [none], proto: ICMP (1), length: 84) n3-lan3 > n0-lan0: ICMP echo reply, id 59466, seq 4, length 64
22:58:29.179217 CDPv2, ttl: 180s, checksum: 692 (unverified), length 241
Device-ID (0x01), length: 11 bytes: 'SAL0739M2PL'
Address (0x02), length: 13 bytes: IPv4 (1) cisco4.isi.deterlab.net
Port-ID (0x03), length: 3 bytes: '9/1'
Capability (0x04), length: 4 bytes: (0x0000002a): Transparent Bridge, L2
Switch, IGMP snooping[icdp]
22:58:29.180136 CDPv2, ttl: 180s, checksum: 692 (unverified), length 241
Device-ID (0x01), length: 11 bytes: 'SAL0739M2PL'
Address (0x02), length: 13 bytes: IPv4 (1) cisco4.isi.deterlab.net
Port-ID (0x03), length: 3 bytes: '9/3'
Capability (0x04), length: 4 bytes: (0x0000002a): Transparent Bridge, L2
Switch, IGMP snooping[icdp]

```

The image shows two terminal windows. The left window, titled 'usersdeterlab.net - PuTTY', shows the output of a ping command from node0 to node2. The output indicates that 7 packets were transmitted and received with 0% packet loss and an average round-trip time of 2.426ms. Below the ping statistics, there are OpenFlow event logs showing packet-in and flow expired events. The right window, titled 'root@nox:/users/sc558ag', shows the output of a tcpdump command on interface nf2c3. The output shows several CDPv2 messages and ICMP echo requests and replies between node0 and node2, with the switch (node0-lan0) acting as the source or destination for the packets.

```

usersdeterlab.net - PuTTY
sc558ag@n0:~$ ping -i 10 -c 7 n2
PING n2-lan2 (10.1.1.7) 56(84) bytes of data.
64 bytes from n2-lan2 (10.1.1.7): icmp_seq=1 ttl=64 time=8.51 ms
64 bytes from n2-lan2 (10.1.1.7): icmp_seq=2 ttl=64 time=2.80 ms
64 bytes from n2-lan2 (10.1.1.7): icmp_seq=3 ttl=64 time=2.86 ms
64 bytes from n2-lan2 (10.1.1.7): icmp_seq=4 ttl=64 time=2.67 ms
64 bytes from n2-lan2 (10.1.1.7): icmp_seq=5 ttl=64 time=2.77 ms
64 bytes from n2-lan2 (10.1.1.7): icmp_seq=6 ttl=64 time=2.42 ms
64 bytes from n2-lan2 (10.1.1.7): icmp_seq=7 ttl=64 time=3.07 ms

--- n2-lan2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 60064ms
rtt min/avg/max/mdev = 2.426/3.590/8.515/2.019 ms
sc558ag@n0:~$

00266/openflow-event[DBG]:received packet-in event from 000000000001 (len:98)
00267/openflow-event[DBG]:received packet-in event from 000000000001 (len:98)
00268/openflow-event[DBG]:received flow expired event from 000000000001

root@nox:/users/sc558ag
-bash-3.2$ sudo tcpdump -vv -i nf2c3
tcpdump: WARNING: nf2c3: no IPv4 address assigned
tcpdump: listening on nf2c3, link-type EN10MB (Ethernet), capture size 96 bytes
22:55:22.359249 CDPv2, ttl: 180s, checksum: 692 (unverified), length 241
Device-ID (0x01), length: 11 bytes: 'SAL0739M2PL'
Address (0x02), length: 13 bytes: IPv4 (1) cisco4.isi.deterlab.net
Port-ID (0x03), length: 3 bytes: '9/3'
Capability (0x04), length: 4 bytes: (0x0000002a): Transparent Bridge, L2
Switch, IGMP snooping[icdp]
22:55:23.359526 CDPv2, ttl: 180s, checksum: 692 (unverified), length 241
Device-ID (0x01), length: 11 bytes: 'SAL0739M2PL'
Address (0x02), length: 13 bytes: IPv4 (1) cisco4.isi.deterlab.net
Port-ID (0x03), length: 3 bytes: '9/4'
Capability (0x04), length: 4 bytes: (0x0000002a): Transparent Bridge, L2
Switch, IGMP snooping[icdp]
22:55:23.396311 CDPv2, ttl: 180s, checksum: 692 (unverified), length 241
Device-ID (0x01), length: 11 bytes: 'SAL0739M2PL'
Address (0x02), length: 13 bytes: IPv4 (1) cisco4.isi.deterlab.net
Port-ID (0x03), length: 3 bytes: '9/2'
Capability (0x04), length: 4 bytes: (0x0000002a): Transparent Bridge, L2
Switch, IGMP snooping[icdp]

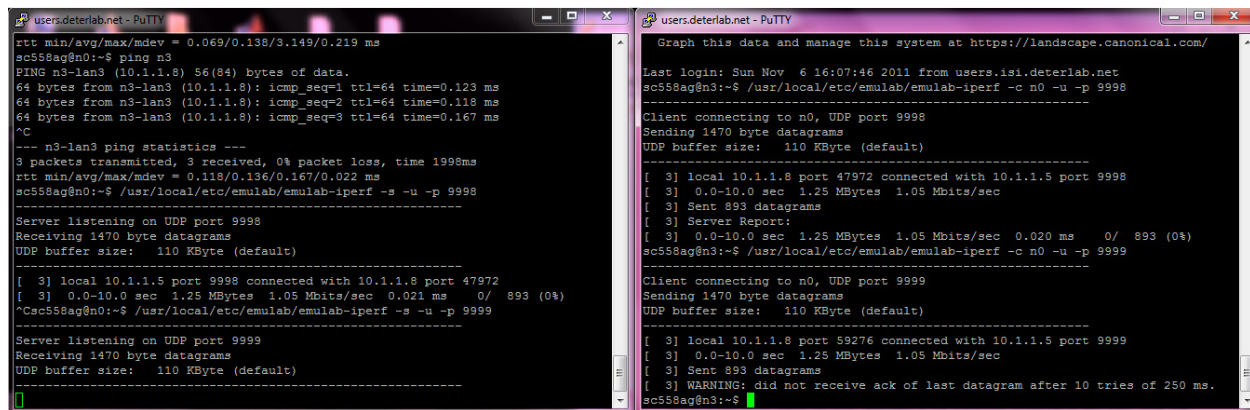
```

## 2. Creating a Firewall on the learning Open Flow Switch

Adding this code to the learning code drops the UDP packet with port destination 9999 to be dropped. When the action mentioned is null, the packet is dropped.

```
if (flow[core.NW_PROTO]==ipv4.ipv4.UDP_PROTOCOL and  
flow[core.TP_DST]==9999) :  
    actions=[]
```

Using iperf tool we tested sending TCP packets and UDP packets via different ports. We observed that using iperf when we sent UDP packets from node3 destined to port 9998 at node0 was received well. And while sending UDP packets from node3 to node0 destined at port 9999 were not received due to dropping. Below is the snapshot for this example.



```
users.deterlab.net - PuTTY
rtt min/avg/max/mdev = 0.069/0.138/3.149/0.219 ms
sc558ag@n0:~$ ping n3
PING n3-lan3 (10.1.1.8) 56(84) bytes of data:
64 bytes from n3-lan3 (10.1.1.8): icmp_seq=1 ttl=64 time=0.123 ms
64 bytes from n3-lan3 (10.1.1.8): icmp_seq=2 ttl=64 time=0.118 ms
64 bytes from n3-lan3 (10.1.1.8): icmp_seq=3 ttl=64 time=0.167 ms
^C
--- n3-lan3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.118/0.136/0.167/0.022 ms
sc558ag@n0:~$ /usr/local/etc/emulab/emulab-iperf -s -u -p 9998
Server listening on UDP port 9998
Receiving 1470 byte datagrams
UDP buffer size: 110 KByte (default)

[ 3] local 10.1.1.5 port 9998 connected with 10.1.1.8 port 47972
[ 3] 0.0-10.0 sec 1.25 MBytes 1.05 Mbits/sec 0.021 ms 0/ 893 (0%)
^C
sc558ag@n0:~$ /usr/local/etc/emulab/emulab-iperf -s -u -p 9999
Server listening on UDP port 9999
Receiving 1470 byte datagrams
UDP buffer size: 110 KByte (default)

[ 3] local 10.1.1.8 port 47972 connected with 10.1.1.5 port 9998
[ 3] 0.0-10.0 sec 1.25 MBytes 1.05 Mbits/sec
[ 3] Sent 893 datagrams
[ 3] Server Report:
[ 3] 0.0-10.0 sec 1.25 MBytes 1.05 Mbits/sec 0.020 ms 0/ 893 (0%)
sc558ag@n3:~$ /usr/local/etc/emulab/emulab-iperf -c n0 -u -p 9999
Client connecting to n0, UDP port 9998
Sending 1470 byte datagrams
UDP buffer size: 110 KByte (default)

[ 3] local 10.1.1.8 port 47972 connected with 10.1.1.5 port 9998
[ 3] 0.0-10.0 sec 1.25 MBytes 1.05 Mbits/sec
[ 3] Sent 893 datagrams
[ 3] Server Report:
[ 3] 0.0-10.0 sec 1.25 MBytes 1.05 Mbits/sec 0.020 ms 0/ 893 (0%)
sc558ag@n3:~$ /usr/local/etc/emulab/emulab-iperf -c n0 -u -p 9999
Client connecting to n0, UDP port 9999
Sending 1470 byte datagrams
UDP buffer size: 110 KByte (default)

[ 3] local 10.1.1.8 port 59276 connected with 10.1.1.5 port 9999
[ 3] 0.0-10.0 sec 1.25 MBytes 1.05 Mbits/sec
[ 3] Sent 893 datagrams
[ 3] WARNING: did not receive ack of last datagram after 10 tries of 250 ms.
sc558ag@n3:~$
```

## References:

1. <http://www.openflow.org/wp/learnmore/>
2. <http://www.orbit-lab.org/wiki/Documentation/Internal/OpenFlow/Notes>
3. <http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf>

## Conclusion:

We were succesful in creating a ‘learning switch’ from the hub using Open Flow switch. We were succesful in creating a firewall for particular type of packets.