CSCI558L Lab9
11/06/2011

ESHA DESAI
SUELA BUZI
LEON ABURIME
HARSH GUPTA

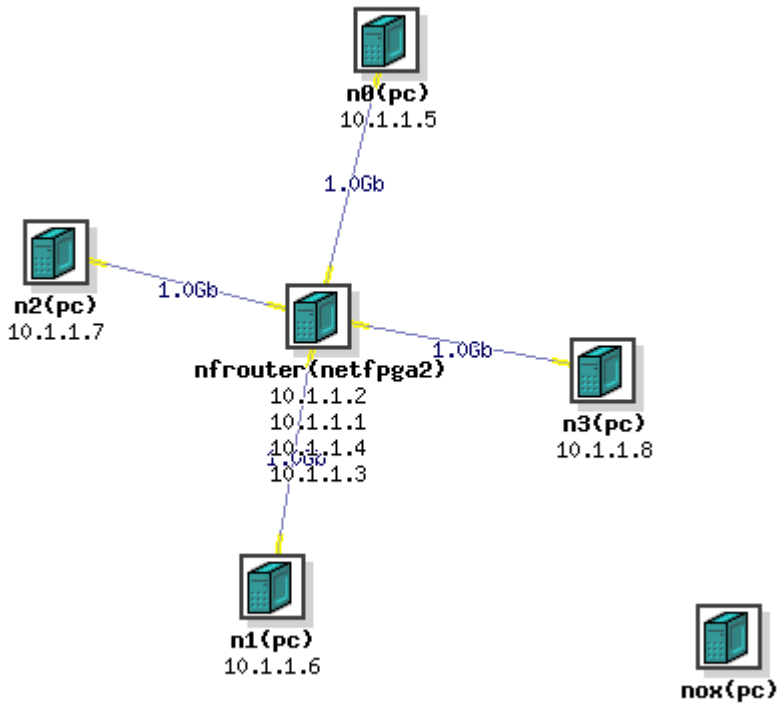
Contents

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=[]
```

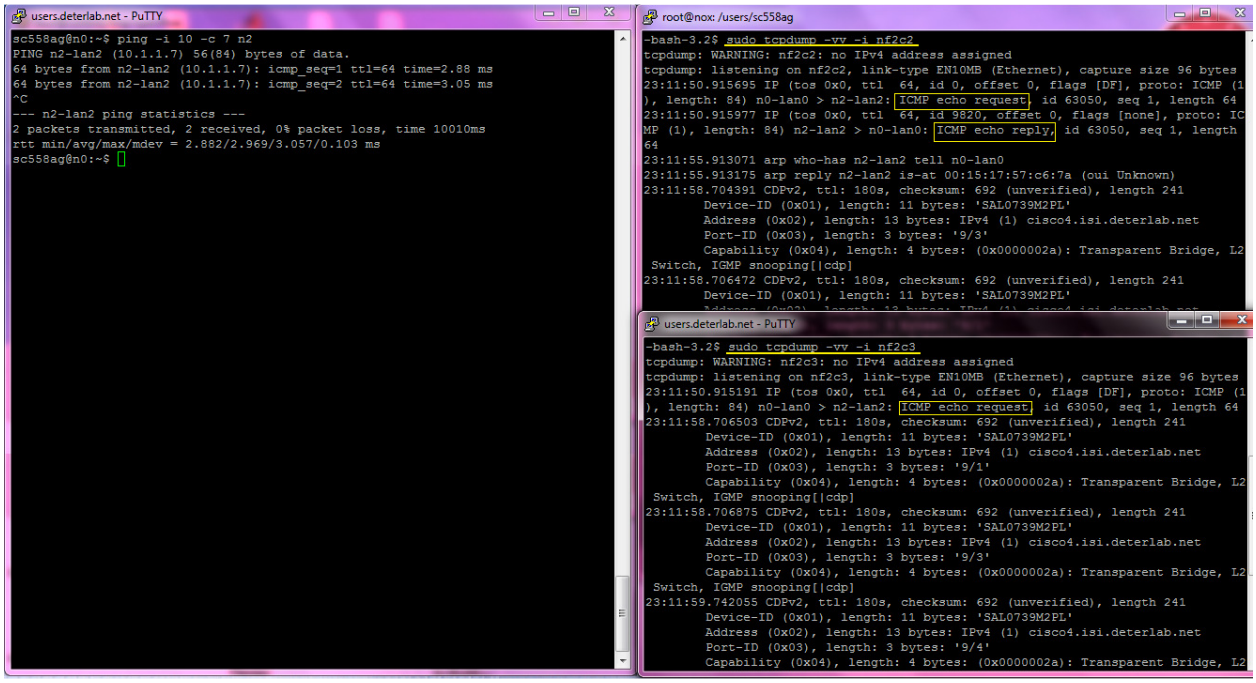
```
            # Later, only after learning controller works:
            # push down flow entry and remove the send_openflow command
above.
            #self.install_datapath_flow ( <fill in params> )
            self.install_datapath_flow(dpid, flow, CACHE_TIMEOUT,
                       openflow.OFP_FLOW_PERMANENT, 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.**

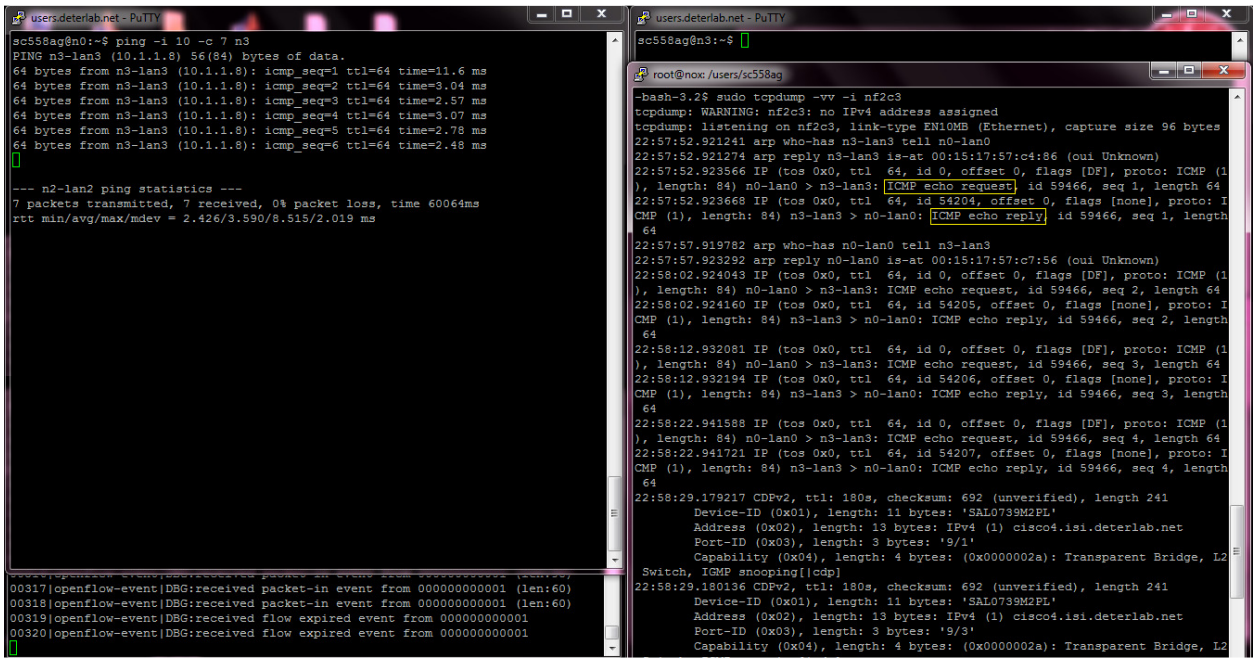
Before the modifications to the code in pyswitch.py, the switch behaved like a hub. We saw that when we tried pinging from node0 to node2, the packet was being received by not only node2 but by all the other nodes as well. All nodes did receive the ICMP packet but only node2 replied back.

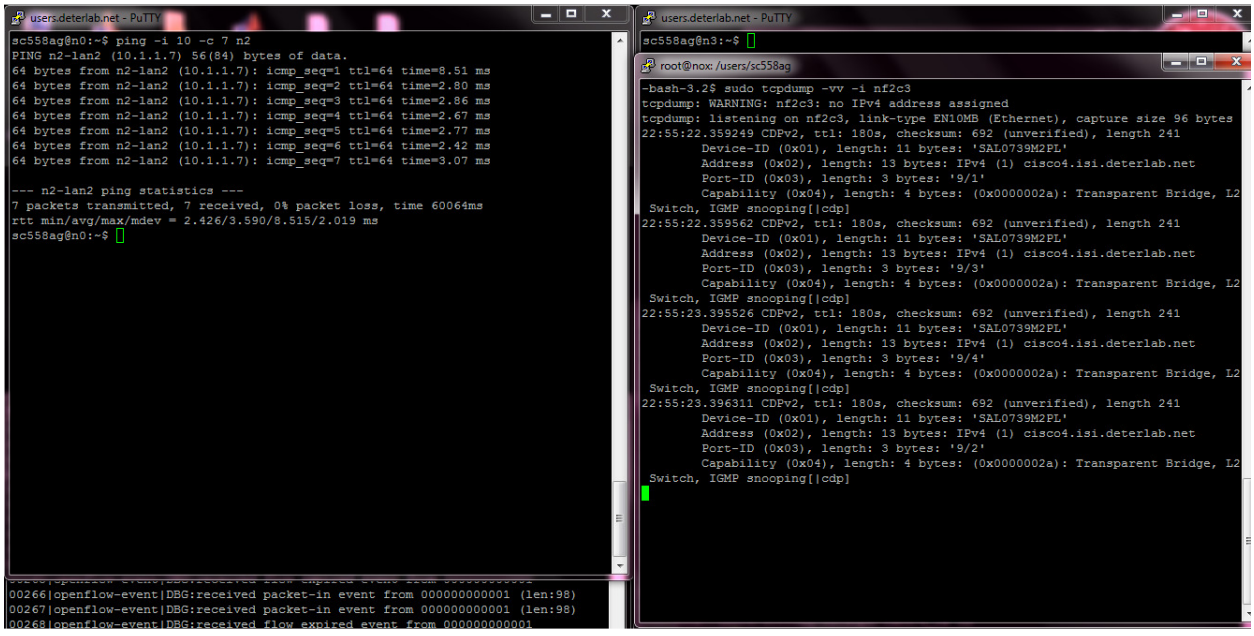
Following is the snapshot of the **tcpdump** results at the interfaces of the switch facing node2 and node3 (Interface names : **nf2c2** and **nf2c3**) :



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 interfaceof 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)



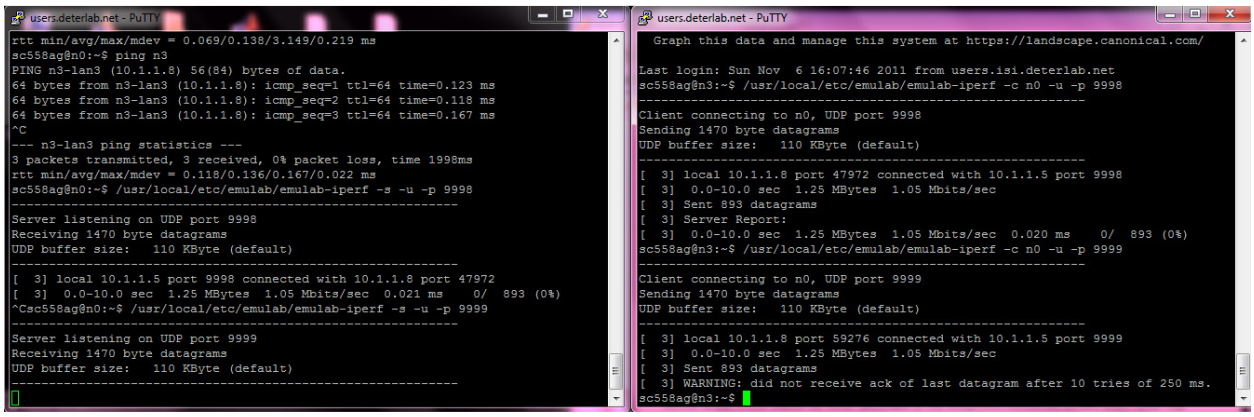


2. **Creating a Firewall on the learning Open Flow Switch**

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.



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 succeful in creating a firewall for particular type of packets.