Building a

Network Training Emulator (NTE)

Software Defined Networking (SDN)

Virtual Machine (VM)

Version 2.0

Diarmuid O'Briain

Table of Contents

Illustration Index

This page is intentionally blank

1. What is NTE-SDN

The Network Training Emulator – Software Defined Networking (NTE-SDN) is a network emulator from **netLabs!UG** Research Centre based on mininet, OpenDaylight and a number of other SDN controllers. Mininet is a development of the ON.lab at Stanford University. Mininet is the ideal tool for experimentation with the OpenFlow protocol and SDN systems. Mininet uses GNU/Linux Containers (LXC) as devices linked by python programs. It is also possible to connect NTE emulated networks to live networks.

2. Install GNU/Linux OS

Install a vanilla Debian GNU/linux 8.5 i386 Operating System. I installed with a default user called **user** who has **sudo** rights.

The following was chosen during the **taskselect** section:

- Debian Desktop environment:.
  - Lightweight X11 Desktop Environment (LXDE).
- SSH Server.
- Standard system utilities.

LXDE was chosen purely for performance reasons. The VMs are expected to run on basic i386 machines.

2.1 Remove Packages

Remove unnecessary packages like **LibreOffice**.

```
user@debian-i386:~$ sudo apt-get purge libreoffice-*
user@debian-i386:~$ sudo apt-get purge alsamixergui
user@debian-i386:~$ sudo apt-get purge mplayer2
user@debian-i386:~$ sudo apt-get purge lxmusic
user@debian-i386:~$ sudo apt-get purge gimp
user@debian-i386:~$ sudo apt-get purge xsane
```

2.2 Change hostname

Change the hostname to **SDN-MN**.

```
user@debian-i386:~$ sudo sed -i.bak 's/debian-i386/SDN-i386/g' /etc/hosts
user@debian-i386:~$ sudo sed -i.bak 's/debian-i386/SDN-i386/g' /etc/hostname
user@debian-i386:~$ cat /etc/hosts
127.0.0.1   localhost
127.0.1.1   SDN-i386.netlabsug.org     SDN-i386

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

user@debian-i386:~$ cat /etc/hostname
SDN-i386
```

2.3 Add user 'sdn'

Create a user **sdn**.

```
user@debian-i386:~$ sudo useradd -m -s /bin/bash sdn
user@debian-i386:~$ sudo passwd sdn
Enter new UNIX password: sdn
Retype new UNIX password: sdn
passwd: password updated successfully

user@debian-i386:~$ sudo usermod -a -G sudo sdn
```

Reboot the VM.

```
user@debian-i386:~$ sudo shutdown -r now
```

2.4 Update sources.list file

Check the file /**etc**/**apt**/**sources.list** file and find the element before **debian.org** as shown.

http://**<ftp.ie>**.debian.org

replace with httpredir which will allow the system to find the nearest repository.

```
user@debian-i386:~$ sudo sed -i.bak 's/ftp.ie/httpredir/g' /etc/apt/sources.list
```

Update the /**etc**/**apt**/**sources.list** file.

```
sdn@SDN-i386:~$ sudo aptitude update
sdn@SDN-i386:~$ sudo aptitude dist-upgrade
```

2.5 Beautify

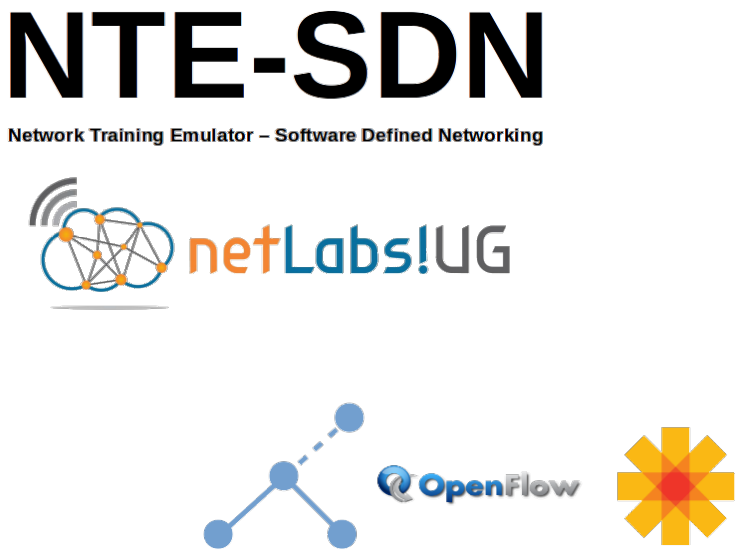

Illustration 1: NTE-SDN Desktop background

Download the following graphic files to user sdn's Pictures directory **/home/sdn/Pictures**.

- http://www.netlabsug.org/NTE/nte-sdn-background.png
- http://www.netlabsug.org/NTE/netLabs!UG-small.png

Right mouse click > Desktop Preferences >

Wallpaper: ~/**Pictures/nte-sdn-background.png**
Background colour: **Preferred colour.**

2.6 Install

Install the following packages.

```
sdn@SDN-i386:~$ sudo aptitude install maven
sdn@SDN-i386:~$ sudo aptitude install openjdk-7-jre
sdn@SDN-i386:~$ sudo aptitude install openjdk-7-jdk
sdn@SDN-i386:~$ sudo aptitude install git
sdn@SDN-i386:~$ sudo aptitude install chromium
sdn@SDN-i386:~$ sudo aptitude install build-essential
sdn@SDN-i386:~$ sudo aptitude install ant
sdn@SDN-i386:~$ sudo aptitude install python-dev
sdn@SDN-i386:~$ sudo aptitude install tcpdump
sdn@SDN-i386:~$ sudo aptitude install mtr
sdn@SDN-i386:~$ sudo aptitude install lynx
sdn@SDN-i386:~$ sudo aptitude install iperf
sdn@SDN-i386:~$ sudo aptitude install tshark
sdn@SDN-i386:~$ sudo aptitude install fping
sdn@SDN-i386:~$ sudo aptitude install wireshark
Should non-Superusers be able to capture packets? <YES>
```

It is assumed python is already installed.

2.7 Wireshark permissions

Add the group **wireshark** to the user **sdn**'s groups, set **dumpcap** within the **wireshark** group and adjust its permissions. Finally set file capabilities on the **dumpcap** file.

```
sdn@SDN-i386:~$ sudo usermod -a -G wireshark sdn
sdn@SDN-i386:~$ sudo chgrp wireshark /usr/bin/dumpcap
sdn@SDN-i386:~$ sudo chmod 750 /usr/bin/dumpcap
sdn@SDN-i386:~$ sudo setcap cap_net_admin,cap_net_raw=eip /usr/bin/dumpcap
sdn@SDN-i386:~$ sudo getcap /usr/bin/dumpcap
/usr/bin/dumpcap = cap_net_admin,cap_net_raw+eip
```

Logout and log back in, confirm the user **sdn** is has **wireshark** as one of its groups.

```
sdn@SDN-i386:~$ id
uid=1001(sdn) gid=1001(sdn) groups=1001(sdn),27(sudo),128(wireshark)
```

3. Mininet

Mininet is a project that creates a virtual network on a computer, a network emulator. With mininet it is possible to develop a network of hosts, switches, routers and links based on a single GNU/Linux kernel. Mininet uses Linux Containers (LXC) lightweight virtualisation to allow experimentation with SDNs and SDN Controllers. For example a SDN Controller can be given a network of devices to work with and because they are based on the GNU/Linux kernel behave exactly as a stand-alone GNU/Linux device.

3.1 Download core files to VM using git

Download the latest version of Mininet.

```
sdn@SDN-i386:~$ cd /tmp
sdn@SDN-i386:/tmp$ git clone git://github.com/mininet/mininet
Cloning into 'mininet'...
remote: Counting objects: 8507, done.
remote: Total 8507 (delta 0), reused 0 (delta 0), pack-reused 8507
Receiving objects: 100% (8507/8507), 2.78 MiB | 464.00 KiB/s, done.
Resolving deltas: 100% (5446/5446), done.
Checking connectivity... done.
sdn@SDN-i386:/tmp$ mv mininet ~/mininet
```

3.2 Install Mininet

Install Mininet, this installs all plus the Wireshark dissector for OpenFlow.

```
sdn@SDN-i386:~$ cd mininet/util
sdn@SDN-i386:~/mininet/util$ ./install.sh -aw
Detected Linux distribution: Debian 8.7 jessie i386
Debian
Installing all packages except for -eix (doxypy, ivs, nox-classic)…
```

During the install an error regarding the *openvswitch-controller* package will appear. The *openvswitch-controller* is not necessary and this error can be safely ignored.

```
E: Unable to locate package openvswitch-controller
Attempting to install openvswitch-testcontroller
Reading package lists...
Building dependency tree...
Reading state information...
E: Unable to locate package openvswitch-testcontroller
Failed - skipping openvswitch-testcontroller
```

Confirm that Mininet is installed OK, test the basic functionality.

```
sdn@SDN-i386:~/mininet/util$ cd ~
sdn@SDN-i386:~$ sudo mn
*** Creating networkc
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

3.3 Upgrading the Mininet Installation

Mininet is user development and therefore new builds are posted to *git* from time to time. To keep the NTE-SDN VM update update Mininet as follows:

```
sdn@SDN-i386:/$ cd ~/mininet/
sdn@SDN-i386:~/mininet$ git fetch
sdn@SDN-i386:~/mininet$ git checkout master
Already on 'master'
Your branch is up-to-date with 'origin/master'.
sdn@SDN-i386:~/mininet$ git pull
Already up-to-date.
sdn@SDN-i386:~/mininet$ sudo make install

sdn@SDN-i386:~/mininet$ nm --version
GNU nm (GNU Binutils for Debian) 2.25
Copyright (C) 2014 Free Software Foundation, Inc.
This program is free software; you may redistribute it under the terms of
the GNU General Public License version 3 or (at your option) any later version.
This program has absolutely no warranty.
```

4. POX Controller

POX is an SDN networking platform written in Python. It is an OpenFlow controller, but can also function as an OpenFlow switch, and can be useful for writing networking software in general. It requires python version 2.7.

```
sdn@SDN-i386:~$ python --version
Python 2.7.9
```

4.1 Installing POX

```
sdn@SDN-i386:~$ git clone http://github.com/noxrepo/pox
Cloning into 'pox'...
remote: Counting objects: 10938, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 10938 (delta 0), reused 0 (delta 0), pack-reused 10936
Receiving objects: 100% (10938/10938), 4.49 MiB | 336.00 KiB/s, done.
Resolving deltas: 100% (6821/6821), done.
Checking connectivity... done.
```

4.2 Running POX

```
sdn@SDN-i386:~$ ./pox/pox.py forwarding.l2_learning
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:core:POX 0.2.0 (carp) is up.
```

4.3 Testing the POX installation

Run a Mininet topology where Floodlight is the SDN controller.

```
sdn@SDN-i386:~$ sudo mn --topo tree,depth=2,fanout=3 --switch ovsk
--controller=remote,ip=127.0.0.1,port=6633 --mac
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(s1, s2) (s1, s3) (s1, s4) (s2, h1) (s2, h2) (s2, h3) (s3, h4) (s3, h5) (s3,
h6) (s4, h7) (s4, h8) (s4, h9)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
```

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9
h2 -> h1 h3 h4 h5 h6 h7 h8 h9
h3 -> h1 h2 h4 h5 h6 h7 h8 h9
h4 -> h1 h2 h3 h5 h6 h7 h8 h9
h5 -> h1 h2 h3 h4 h6 h7 h8 h9
h6 -> h1 h2 h3 h4 h5 h7 h8 h9
h7 -> h1 h2 h3 h4 h5 h6 h8 h9
h8 -> h1 h2 h3 h4 h5 h6 h7 h9
h9 -> h1 h2 h3 h4 h5 h6 h7 h8
*** Results: 0% dropped (72/72 received)
mininet>
```

5. Project Floodlight

The Project Floodlight Open SDN Controller operates with OpenFlow switches. It is an enterprise class, Apache licensed, Java based SDN Controller and has its origins with Big Switch Networks.

5.1 Install Java 8

Project Floodlight requires Java8 to build and Debian defaults to OpenJDK 7.

```
sdn@SDN-i386:~$ java -version
java version "1.7.0_121"
OpenJDK Runtime Environment (IcedTea 2.6.8) (7u121-2.6.8-2~deb8u1)
OpenJDK Client VM (build 24.121-b00, mixed mode, sharing)
```

Install Oracle Java8.

```
sdn@SDN-i386:~$ sudo -s
root@SDN-i386:~# echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu
xenial main" | tee /etc/apt/sources.list.d/webupd8team-java.list
root@SDN-i386:~# echo "deb-src
http://ppa.launchpad.net/webupd8team/java/ubuntu xenial main" | tee -a
/etc/apt/sources.list.d/webupd8team-java.list
root@SDN-i386:~# apt-key adv --keyserver hkp://keyserver.ubuntu.com:80
--recv-keys EEA14886
root@SDN-i386:~# apt-get update
root@SDN-i386:~# apt-get install oracle-java8-installer
root@SDN-i386:~# apt install oracle-java8-set-default
root@SDN-i386:~# exit
sdn@SDN-i386:~$ java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) Client VM (build 25.121-b13, mixed mode)
```

5.2 Download from github

Floodlight is simple to download from Github and build using Apache *ant*, a Java based build tool. Ant's build files are written in XML and they take advantage of being open standard, portable and easy to understand.

```
sdn@SDN-i386:~$ cd /tmp
sdn@SDN-i386:/tmp$ git clone git://github.com/floodlight/floodlight.git
sdn@SDN-i386:/tmp$ mv floodlight ~/floodlight
sdn@SDN-i386:/tmp$ cd ~/floodlight
sdn@SDN-i386:~/floodlight$ ant
BUILD SUCCESSFUL
Total time: 34 seconds

sdn@SDN-i386:~$ sudo mkdir /var/lib/floodlight
sdn@SDN-i386:~$ sudo chmod 777 /var/lib/floodlight
```

5.3 Running Floodlight

Run the *floodlight.jar* file produced by *ant*.

```
sdn@SDN-i386:~$ cd ~/floodlight
sdn@SDN-i386:/floodlight$ java -jar ./target/floodlight.jar
```

Floodlight will start running and print debug output to your console.

5.4 Testing the Floodlight installation

Run a Mininet topology where Floodlight is the SDN controller.

```
sdn@SDN-i386:~$ sudo mn --topo tree,depth=2,fanout=3 --switch ovsk
--controller=remote,ip=127.0.0.1,port=6653 --mac
[sudo] password for sdn:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(s1, s2) (s1, s3) (s1, s4) (s2, h1) (s2, h2) (s2, h3) (s3, h4) (s3, h5) (s3,
h6) (s4, h7) (s4, h8) (s4, h9)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:

mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9
h2 -> h1 h3 h4 h5 h6 h7 h8 h9
h3 -> h1 h2 h4 h5 h6 h7 h8 h9
h4 -> h1 h2 h3 h5 h6 h7 h8 h9
h5 -> h1 h2 h3 h4 h6 h7 h8 h9
h6 -> h1 h2 h3 h4 h5 h7 h8 h9
h7 -> h1 h2 h3 h4 h5 h6 h8 h9
h8 -> h1 h2 h3 h4 h5 h6 h7 h9
h9 -> h1 h2 h3 h4 h5 h6 h7 h8
*** Results: 0% dropped (72/72 received)
```

6. OpenDaylight

Project OpenDaylight is a Linux Foundation Collaborative Project. The software combines SDN components including a fully pluggable controller, interfaces, protocol plug-ins and applications to create a framework for SDN and Network Functions Virtualisation (NFV) solutions. The current release is Boron.

6.1 Install OpenDaylight

Download the latest build of OpenDaylight (ODL) from:

https://www.opendaylight.org/downloads

In this version of NTE-SDN the Boron version of ODL is deployed.

Copy the download to the /**tmp** directory on the NTE-SDN VM.

```
sdn@SDN-i386:~$ ls /tmp
distribution-karaf-0.5.2-Boron-SR2.tar.gz
sdn@SDN-i386:~$ cd /tmp
sdn@SDN-i386:/tmp$ tar -xzvf distribution-karaf-0.5.2-Boron-SR2.tar.gz
sdn@SDN-i386:/tmp$ mv distribution-karaf-0.5.2-Boron-SR2.tar.gz ~/odl
```

Create JAVA_HOME.

```
sdn@SDN-i386:~$ cat << JAVARC >> ~/.bashrc
#JAVA_HOME for OpenDaylight
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
JAVARC

sdn@SDN-i386:~$ export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

6.2 Running ODL karaf

ODL uses Apache Karaf which is a small Open Services Gateway initiative (OSGi) based runtime which provides a lightweight container onto which various components and applications can be deployed. Karaf provides an ecosystem for ODL.

To run karaf:

```
sdn@SDN-i386:~$ ./odl/bin/karaf
```

```
        _________                    ________                .__  .__       .__     __
        \_____  \ ______   ____   ____ \_____ \ _____  ___.__.|  | |__| ____ |  |___/  |_
         /   |   \\____ \_/ __ \ /    \ |    |  \\__  \<   |  ||  | |  |/ ___\|  |  \   __\
        /    |    \  |_> >  ___/|   |  \|    `   \/ __ \\___  ||  |_|  / /_/  >   Y  \  |
        \_______  /   __/ \___  >___|  /_______  (____  / ____||____/__\___  /|___|  /__|
                \/|__|        \/     \/        \/     \/\/           /_____/      \/
```

```
Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

opendaylight-user@root>

* Press *tab* for a list of available commands.
* Typing *[cmd] --help* will show help for a specific command.
* Press *ctrl-d* or type *system:shutdown* or *logout* to shutdown OpenDaylight.

opendaylight-user@root> version
3.0.7
```

6.3 Installing openDaylight User eXperience (DULX)

On the *karaf* shell install DLUX.

- *odl-restconf* – Representational STate (REST) like protocol that provides a programmatic interface over Hyper Text Transfer Protocol (HTTP) for accessing data on port 8080 for HTTP requests.
- *odl-l2switch-all* – Layer2 switch functionality.
- *odl-mdsal-apidocs* - Model Driven Service Abstraction Layer (MD-SAL) Application Programmable Interface (API) Documentation. They can be accessed at: http://<IP addr>:8181/apidoc/explorer/index.html.
- *odl-dlux-all* - Graphical user interface for OpenDaylight based on the AngularJS framework.

```
opendaylight-user@root> feature:install odl-restconf
opendaylight-user@root> feature:install odl-l2switch-all
opendaylight-user@root> feature:install odl-mdsal-apidocs
opendaylight-user@root> feature:install odl-dlux-all
```

Features can be installed in one command like this also.

```
opendaylight-user@root> feature:install odl-restconf odl-l2switch-all odl-
mdsal-apidocs odl-dlux-all
```

6.4 Testing the ODL installation

Run a *Mininet* topology where the ODL is the SDN controller.

```
sdn@SDN-i386:~$ sudo mn --topo tree,depth=2,fanout=3 --switch ovsk
--controller=remote,ip=127.0.0.1,port=6633 --mac
[sudo] password for sdn:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(s1, s2) (s1, s3) (s1, s4) (s2, h1) (s2, h2) (s2, h3) (s3, h4) (s3, h5) (s3,
h6) (s4, h7) (s4, h8) (s4, h9)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:

mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9
h2 -> h1 h3 h4 h5 h6 h7 h8 h9
h3 -> h1 h2 h4 h5 h6 h7 h8 h9
h4 -> h1 h2 h3 h5 h6 h7 h8 h9
h5 -> h1 h2 h3 h4 h6 h7 h8 h9
h6 -> h1 h2 h3 h4 h5 h7 h8 h9
h7 -> h1 h2 h3 h4 h5 h6 h8 h9
h8 -> h1 h2 h3 h4 h5 h6 h7 h9
h9 -> h1 h2 h3 h4 h5 h6 h7 h8
*** Results: 0% dropped (72/72 received)
```

6.5 DLUX User interface

In the ODL user interface the *Mininet* network should be visible.

Use Google Chrome of Chromium browser.

http://<NTE-SDN-IP-Addr>:8181/index.html

Login: **admin**
Password: **admin**

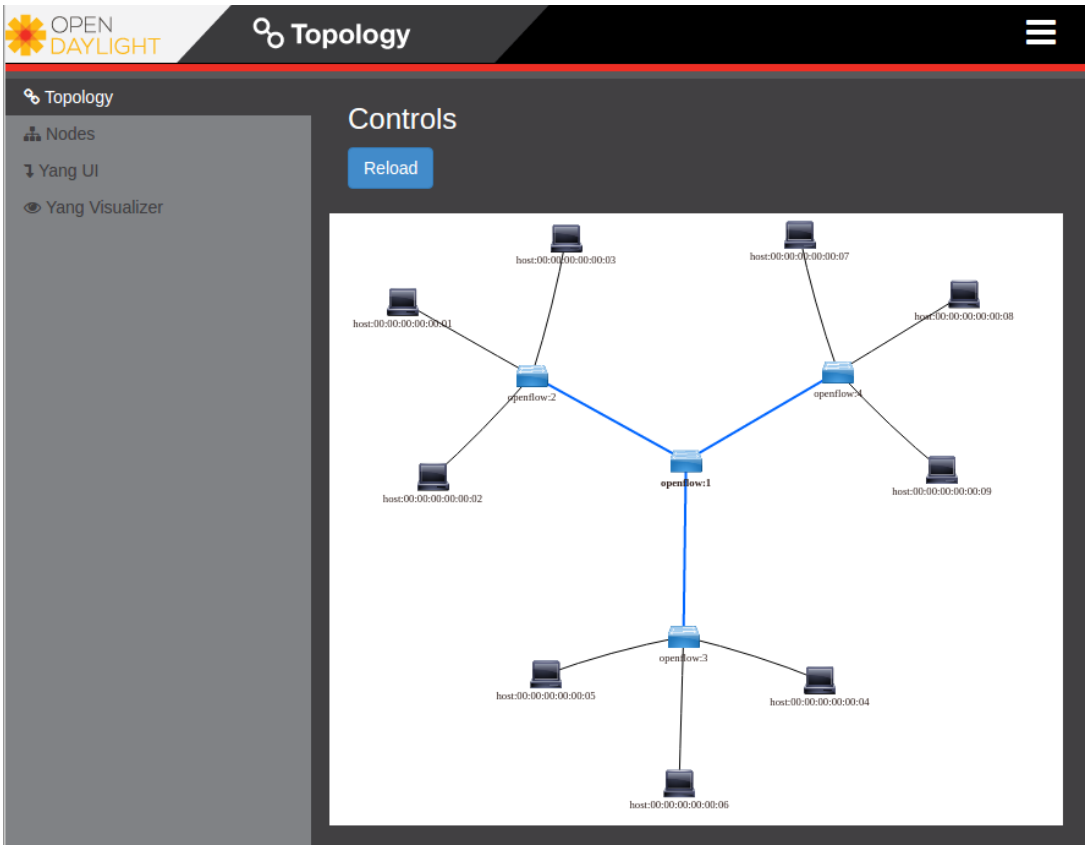

Illustration 2: DLUX Topology

7. Monitoring with Wireshark

Monitoring the OpenFlow traffic with Wireshark is accomplished by monitoring the localhost interface. In all three SDN Controller options traffic between the Mininet network and the controller was via 127.0.0.1, the localhost. Using a filter ***openflow_v1*** will show the traffic between controller and the Open virtual Switch (OvS).

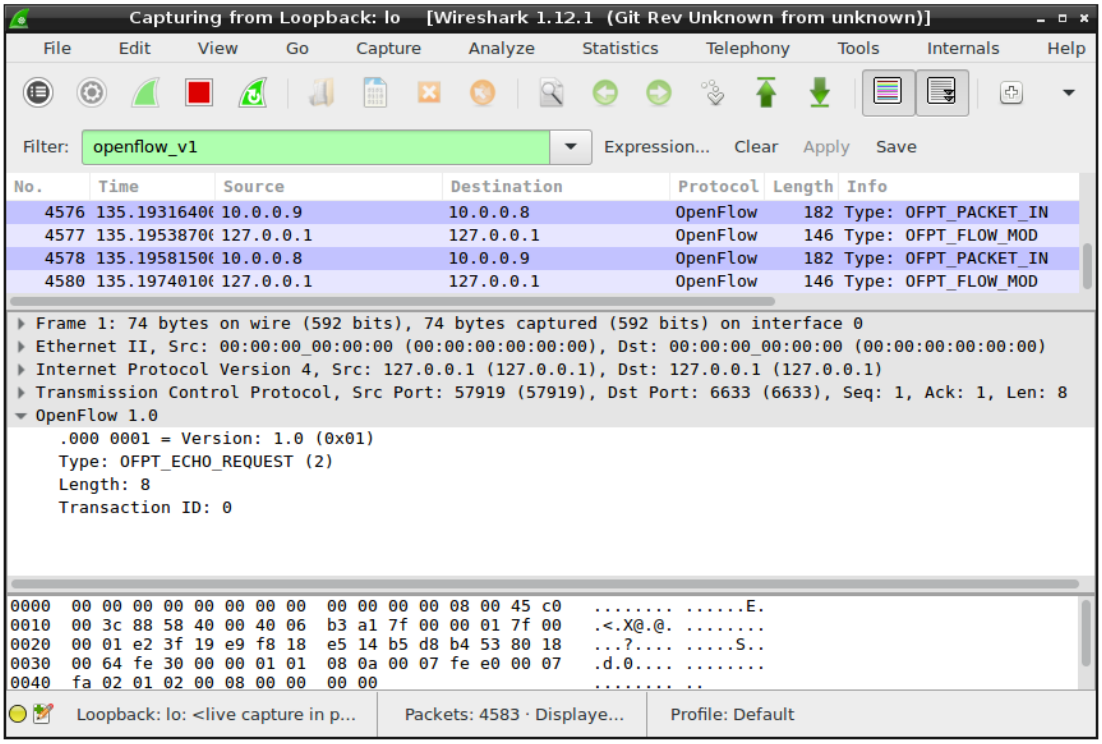

Illustration 3: Wireshark capture