EFFICIENT TRAFFIC MANAGEMENT IN SOFTWARE-DEFINED NETWORKING

TUSHAR AHERKAR <h20171030084@hyderabad.bits-pilani.ac.in>
DAMINI NANAWARE <h20171030088@hyderabad.bits-pilani.ac.in>
SOURABH SETHI <h20171030072@hyderabad.bits-pilani.ac.in>

1 Introduction

SDN is intended to address the way that the static architecture of conventional networks doesn‘t support the dynamic, scalable computing and storage needs of more present day computing environments such as data centers. This is done by decoupling or disassociating the system that makes decisions about where traffic is sent (the control plane) from the underlying systems that forward traffic to the intended destination (the data plane). The Software Defined Networking method centralizes control of the network by separating the control logic to off-device computer resources.

An SDN controller is an application in software-defined networking that manages flow control to enable intelligent networking. SDN controllers such as OpenDayLight are based on protocols (OpenFlow) , that allows servers to tell switches where to send packets.

OpenDaylight: OpenDaylight is an exceedingly accessible, measured, extensible, versatile and multi-convention controller framework worked for SDN arrangements on current heterogeneous multivendor systems.

OpenDaylight gives a model-driven service abstraction platform that enables clients to compose applications that effectively work over a wide range of hardware and southbound APIs. These applications utilize the controller to accumulate network intelligence, run algorithms to perform some analysis, and afterward utilize the controller to arrange the new rules, if any, throughout the network.

Mininet Emulator: Mininet is a product emulator for prototyping a large network on to a single machine. Mininet emulator is a cheap and rapidly configurable system testbed. It enables the client to rapidly make, cooperate with, customize and share a software-defined network (SDN) prototype to simulate a network.

Mininet utilizes virtual hosts, switches, and links to create a system on a single OS kernel, and uses the real network stack to process packets and connect to real networks. Moreover, Unix/Linux-based applications are likewise permitted to keep running on virtual hosts. In an OpenDayLight arrangement simulated by Mininet, a real OpenDayLight controller application can keep running on another machine or on a similar machine where virtual hosts are emulated.

2 Project Objective

The primary aim of mid semester demo is to understand Software Defined Networks and usage of Mininet and OpenDayLight controller. Along with this we aim to set up a simple topology and simulate connection between the nodes in the topology. Simulation is performed in using Mininet and traffic management based on the topology is done using OpenDayLight controller. This will act as a stepping stone towards our final project demo.

3 Tools used and Installation overview :

Two tools are mainly used till now in the project to create the SDN environment. These are :

a . Mininet

b. OpenDayLight

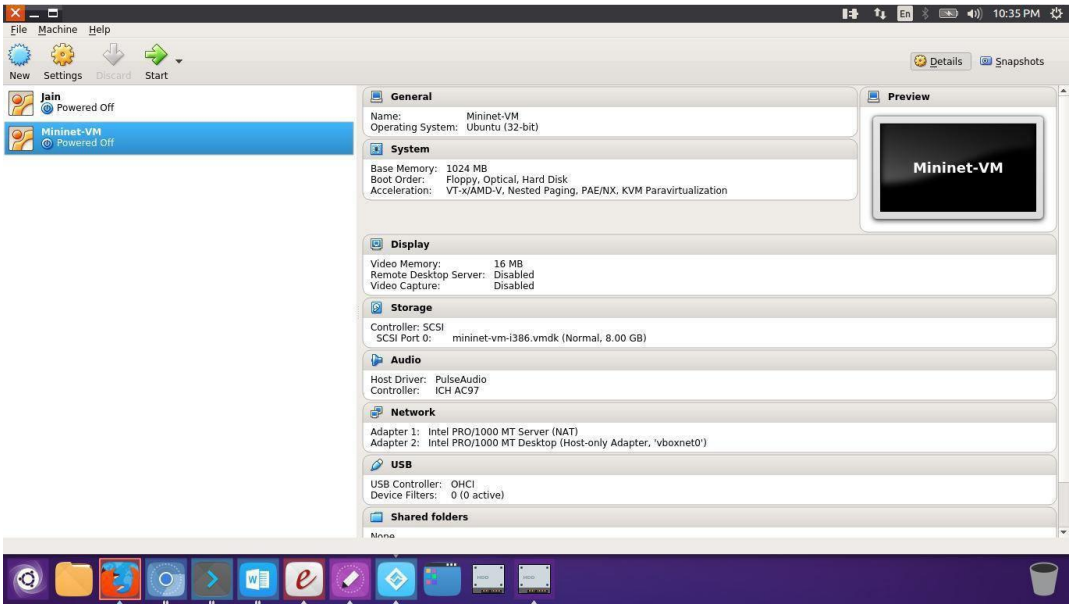
Mininet - Mininet is a network simulator used to create scalable SDNs using Linux processes. Mininet is used in this project to simulate the topology and test the traffic flows.

OpenDayLight – Topology created using mininet emulator is viewed using OpenDayLight controller. All the details regarding host, switches and their variation is customized using this software controller.

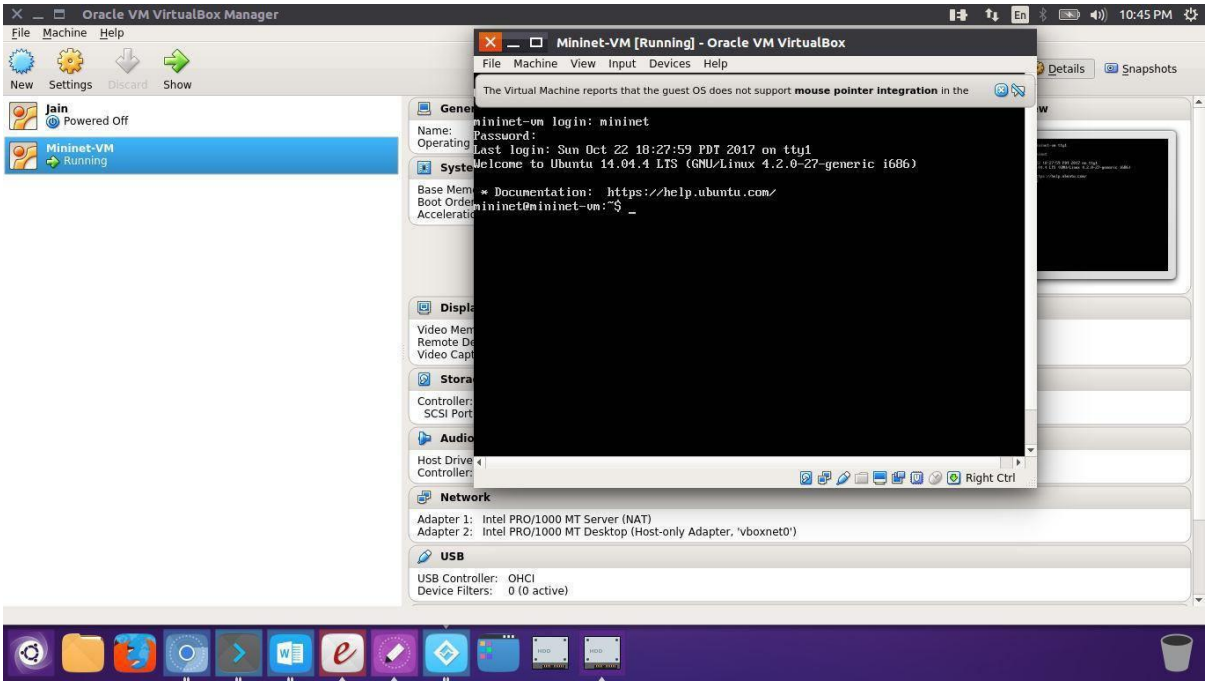
Installation of mininet guide :

We have installed a mininet virtualbox over oracle virtual box . Reference Link is shared at the end [1] where detailed commands being used are mentioned. Following are the steps followed to install mininet virtualbox :

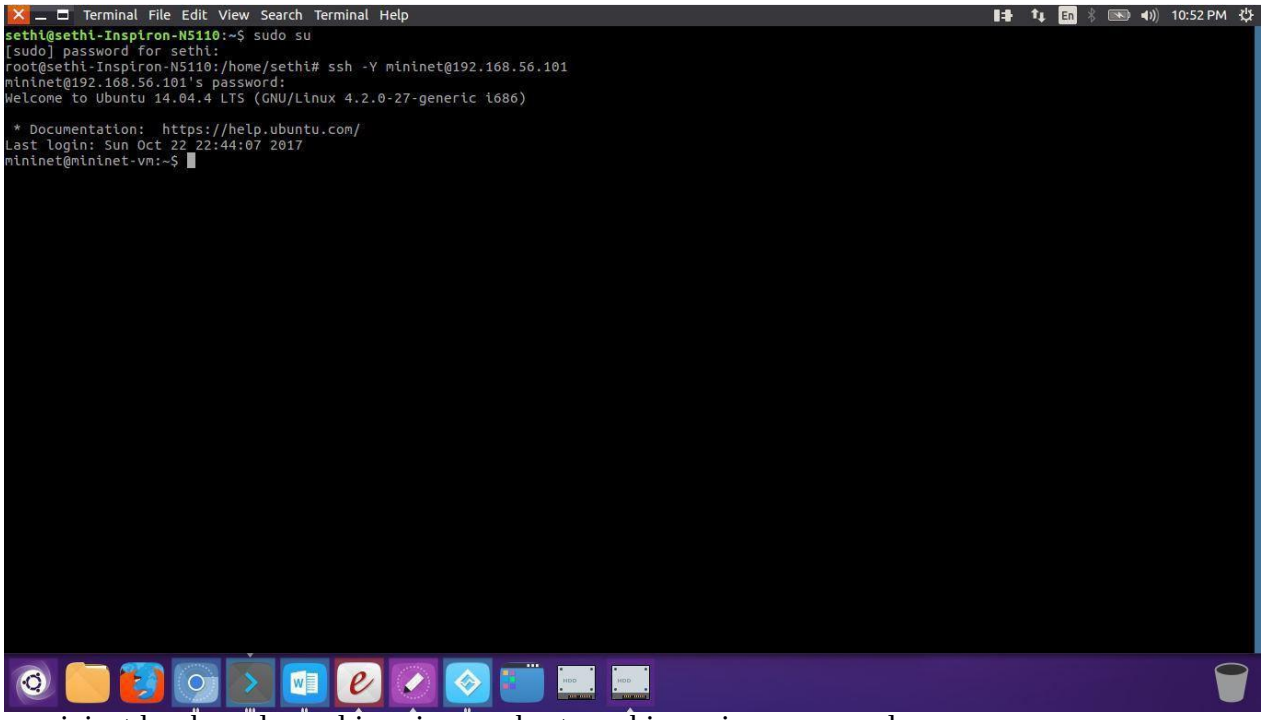
1. Downloaded the Mininet virtual machine.
2. Import the virtual machine into VirtualBox
3. Add a Host-only Adapter in VirtualBox
4. Add Network Adapter to Mininet virtual machine
5. Start the Mininet virtual machine



6. Once started , Login ID and password are required which are “mininet” by default as shown below are used :



7.Using SSH to connect to the Mininet VM as shown below from host machine :



Here mininet has been logged in using our host machine using command
ssh -Y mininet@192.168.56.101

192.168.56.101 is the mininet Virtual machine IP .

Hence , by now , our emulator has been installed successfully .

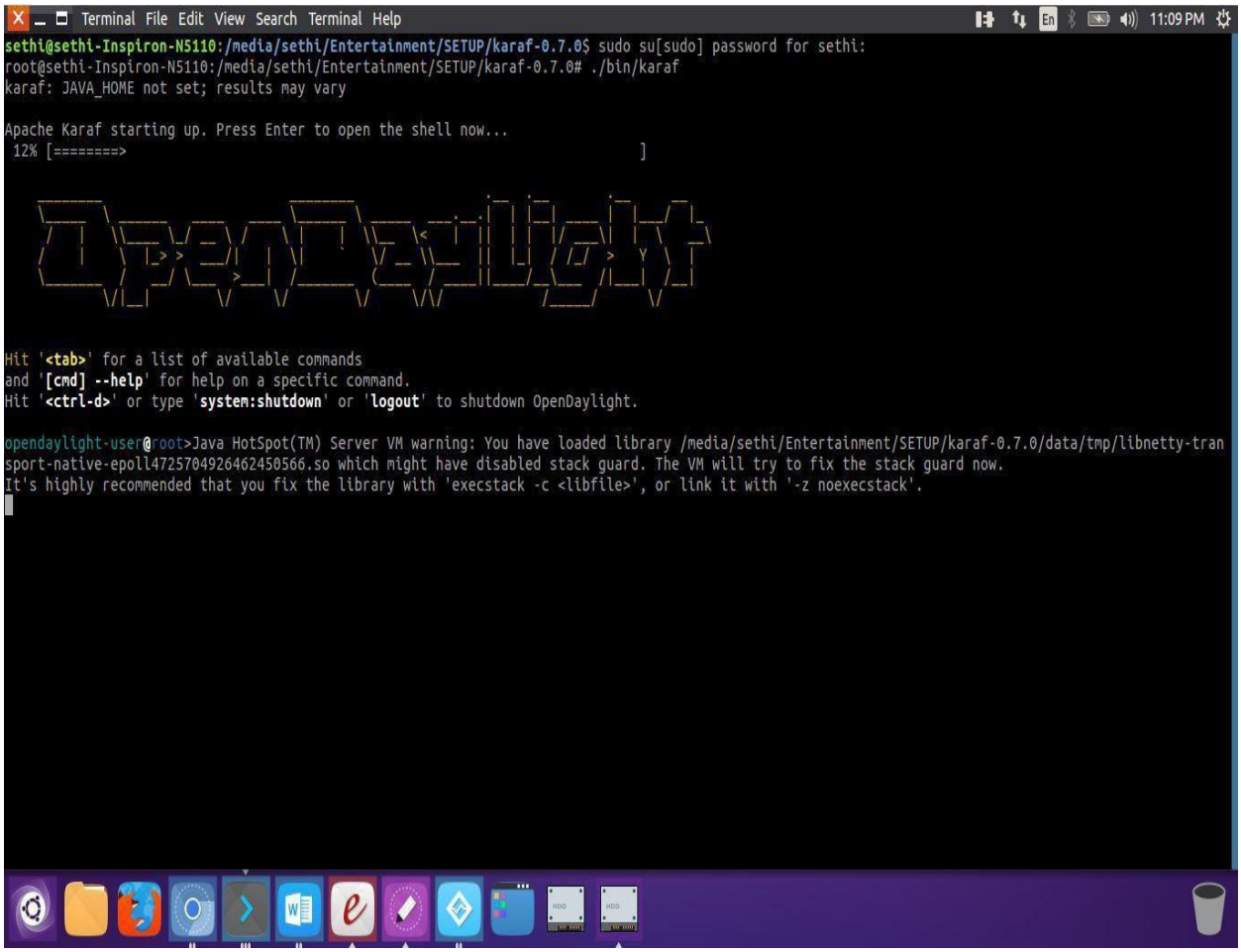
Installation of OpenDayLight guide :

We have installed a OpenDayLight controller for observing the network topology created. Reference Link is shared at the end [2] in reference section where detailed commands being used are mentioned.

Following are the steps followed to install OpenDayLight controller :

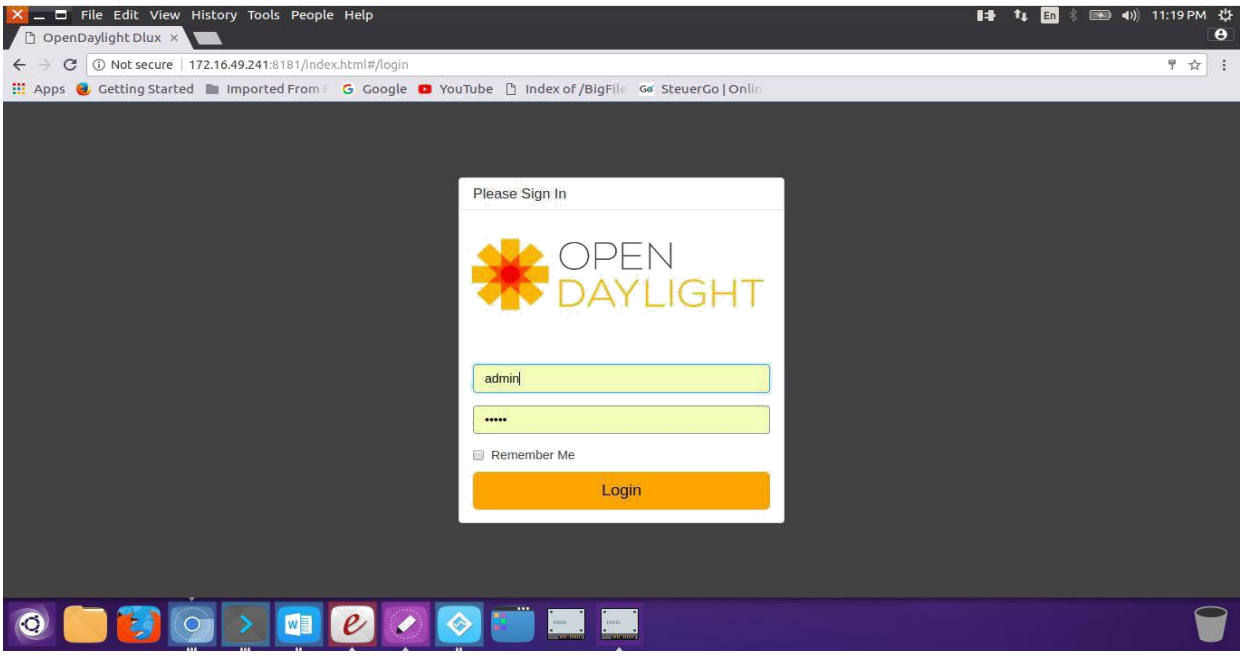
1. Downloaded the OpenDayLight latest version (Nitrogen)from https://www.opendaylight.org/
2. Installed OpenDaylight using the steps shared in reference link[2]
3. Start OpenDaylightas shown below using command :

./bin/karaf (In the karaf directory path where files are downloaded)



4. Now run the web application of OpenDayLight from our browser using below mentioned link http://172.16.49.241:8181/index.html#/node/index

Here , 172.16.49.241 is IP of machine where controller has been installed



Hence the environmental requirements for creating the topology has been successfully done .

4 Methodology

Now that we have finished installation let, we can start with first step of implementation:

4.1 Building a topology on Mininet

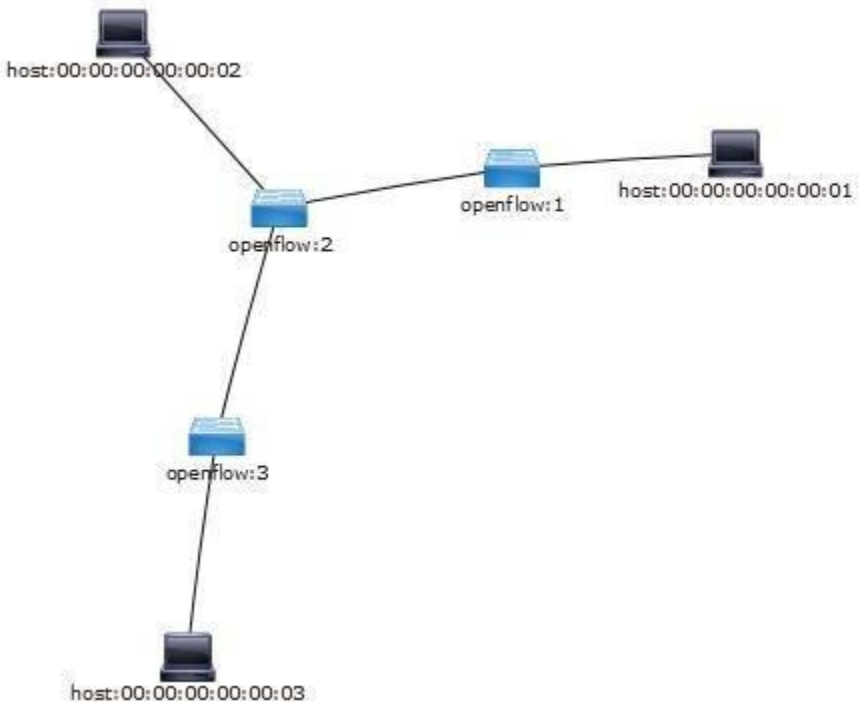
1. Let us consider a simple linear topology of three switches
2. Each switch is connected to one host
3. The MAC address on each host can be set as a simple number
4. The remote controller, OpenDaylight, is at IP address 192.168.217.135:6633

Following image shows the command used in order to create the linear topology.

The default topology is the **minimal** topology, which includes one OpenFlow kernel switch connected to two hosts, plus the OpenFlow reference controller. This topology could also be specified on the command line with–**topo minimal**. Here we chose **linear** topology.

```
tushar@ubuntu:~$ sudo mn --topo linear,3 --mac --controller=remote,ip=192.168.217.135,port=6633 --switch ovsk,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s1, s2) (s2, s3)
*** Configuring hosts
h1 h2 h3
*** Starting controller
*** Starting 3 switches
s1 s2 s3
*** Starting CLI:
```

We can now see this topology in our OpenDaylight interface. We can see three nodes and Openflow switches connected to a OpenDaylight controller.



4.2 Testing the created network:

Now to test our connections we ping all the created nodes. Every host should be able to reach every other host.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

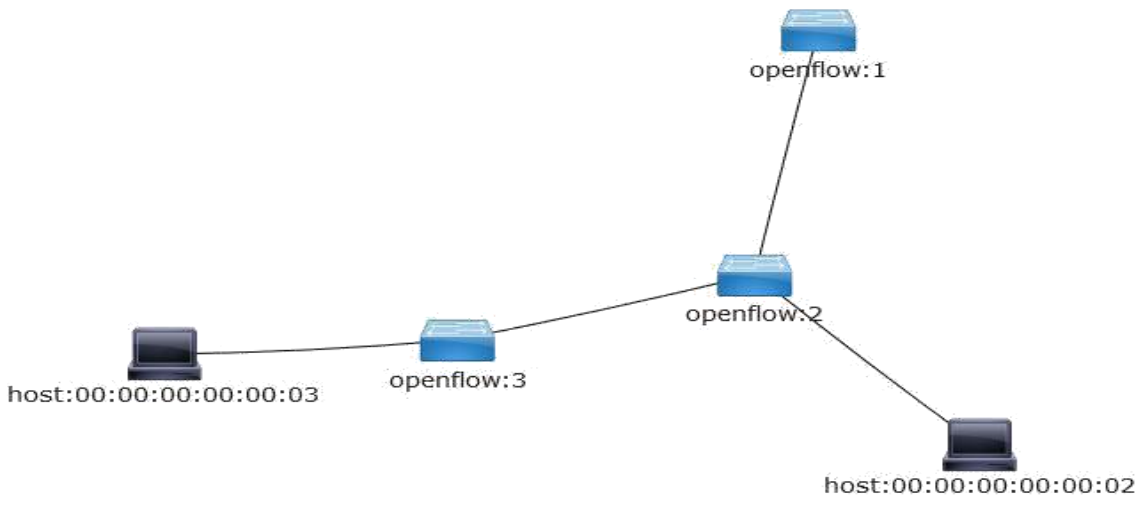
Next we can try different commands to check number of nodes

```
mininet> nodes
available nodes are:
c0 h1 h2 h3 s1 s2 s3
```

Now we experimented to turn down a link and then tried pinging the nodes. We turn down link between h1 and s1 using following command and then ping all.

```
mininet> link h1 s1 down
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X h3
h3 -> X h2
*** Results: 66% dropped (2/6 received)
```

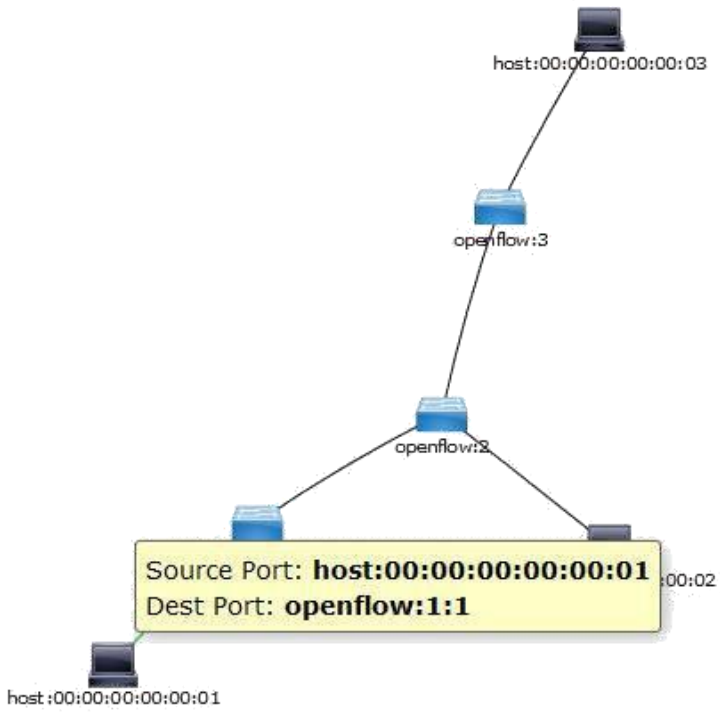
As we can see we now cannot reach all the from h1 and cannot reach h1 through h2 and h3. Below figure shows view in the OpenDaylight interface where we can see h1 is not pre



We can again connect this link and then check pinging the nodes.

```
mininet> link h1 s1 up
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

Following are the results shown in the OpenDaylight interface



The above figure shows host h1 and link between h1 and s1 again being connected.

5 Conclusion

We studied Software Defined Networks thoroughly. We were able to understand and set up a topology using mininet and OpenDaylight controller. We established a connection between these nodes which establishes our mid semester progress. We plan to manage and shape traffic flow by next few weeks which completes our project statement.

6 References:

[1] http://www.brianlinkletter.com/set-up-mininet/

[2] http://www.brianlinkletter.com/using-the-opendaylight-sdn-controller-with-the-mininet-network-emulator/