PUNE INSTITUTE OF COMPUTER TECHNOLOGY,

DHANKAWADI PUNE-43.

A

Seminar Report

On

SOFTWARE-DEFINED NETWORKING using OpenFlow protocol v1.3

SUBMITTED BY

NAME: Pranav Shailendra Deshpande

ROLL NO: 3315

CLASS: TE III

GUIDED BY

PROF. Sumit Shinde

ISO 9001 : 2008 Certified

COMPUTER ENGINEERING DEPARTMENT

Academic Year:2016-17

PUNE INSTITUTE OF COMPUTER TECHNOLOGY,

DHANKAWADI PUNE-43.

CERTIFICATE

ISO 9001 : 2008 Certified

This is to certify that Mr. Pranav Shailendra Deshpande

RollNo. 3315 a student of T.E. (Computer Engineering Department) Batch 2016-2017, has satisfactorily completed a seminar report on "Software-Defined Networking using OpenFlow Protocol v1.3" under the guidance of prof. Sumit Shinde towards the partial fulfillment of the third year Computer Engineering Semester II of Pune University.

_____ _____

Prof. Sumit Shinde
Internal Guide

Dr. R.B. Ingle
Head of Department,
Computer Engineering

Date:-

Place:- PICT, Pune.

Software-Defined Networking Using OpenFlow Protocol (v1.3)

Abstract:

We explain the notion of software-defined networking(SDN), whose southbound interface may be implemented by the OpenFlow protocol. Describing the operation of OpenFlow and summarize the features of specification versions 1.3.

Presenting overview of existing SDN-based applications grouped by topic areas. Implementation of a demonstrative Network will be done using a SDN specialized tool GNS3 including ODL as controller, OpenVSwitch switches using OpenFlow protocol 1.3.

Keywords*: Software-defined networking; OpenFlow; OpenVSwitch; Network applications.*

INTRODUCTION

Software-defined networking(SDN)has gained a lot of attention in recent years, because it addresses the lack of programmability in existing networking architectures and enables easier and faster network innovation. SDN clearly separates the data plane from the control plane and facilitates software implementations of complex networking applications on top. There is the hope for less specific and cheaper hardware that can be controlled by software applications through standardized interfaces. Additionally, there is the expectation for more flexibility by dynamically adding new features to the network in the form of networking applications. This concept is known from mobile phone operating systems, such as Apple's iOS and Google's Android, where "apps" can dynamically be added to the system. SDN using the definition from the Open Networking Foundation (ONF).

OpenFlow is just an option for a control protocol in SDN, but it is the predominant one. OpenFlow is the first standard communications interface defined between the control and forwarding layers of an SDN architecture. OpenFlow allows direct access to and manipulation of the forwarding plane of network devices such as switches and routers, both physical and virtual (hypervisor-based).OpenFlow-based SDN technologies enable IT to address the high-bandwidth, dynamic nature of today's applications, adapt the network to ever-changing business needs, and significantly reduce operations and management complexity.

The docker support is a new feature of GNS3 1.5. This features has been started by Goran Cetusic during the Google Summer Of Code and finished by the GNS3 core team.Containers use the host kernel this mean they consume, less less RAM and CPU.

Challenges faced are Addressing dynamic real-time change, accommodating rapid on-demand growth, Integrating service context

The programmability and centralized control of the network topology in SDN allow enterprises to incorporate various applications easily to improve efficiency, reduce complexity, streamline processes, and provide superior user experience.
The need of securing the network from malicious attacks as well as from viruses and worms lead to the introduction of firewalls in software defined networks.

PROPOSED MATHEMATICAL MODEL

MATHEMATICAL MODEL:

Let S be the System where,

S= {s, c, sw, comp, funt,y,DD,NDD}
Where,
s = start state
c = Open Daylight controller
c = {lc, ip, port, auth, rel}

where,
lc =Load Connection, ip= {server IP address}, port=6633, auth= {uname, pass}, rel= {reload topology}

I = Initiate Open Flow protocol v1.3 along topology.

h = Handle up connections.
Where, h = {uip, fip, ovs1, ovs2, ovs3, ovs4}
Where, elements in h are IP Addresses

Comp= {us, ft,h1,natC,OvS[1-4]}
Where, us=Ubuntu Server, ft=Firefox Terminal, natC=Nat Cloud, OvS= 4 Open vSwitches

Fuct={assignIP,ping1,ping2,ping3,ping4,ping5}
Where,

AssignIP={DHCP/Static IP addresses to Server,Firefox Utility,Ovs[1-4]
Ping1=Ping from Firefox terminal to Ubuntu Server
Ping2= Ping from OvS1 to Ubuntu Server
Ping3= Ping from OvS2 to Ubuntu Server
Ping4= Ping from OvS3 to Ubuntu Server
Ping5= Ping from OvS4 to Ubuntu Server

Let Y be the set of outputs
Where,

Y={successPing,ODLSwitchDisp}
SuccessPing= ping between the switches is successful.
ODLSwitchDisp= The UI of ODL shows the connected switches.

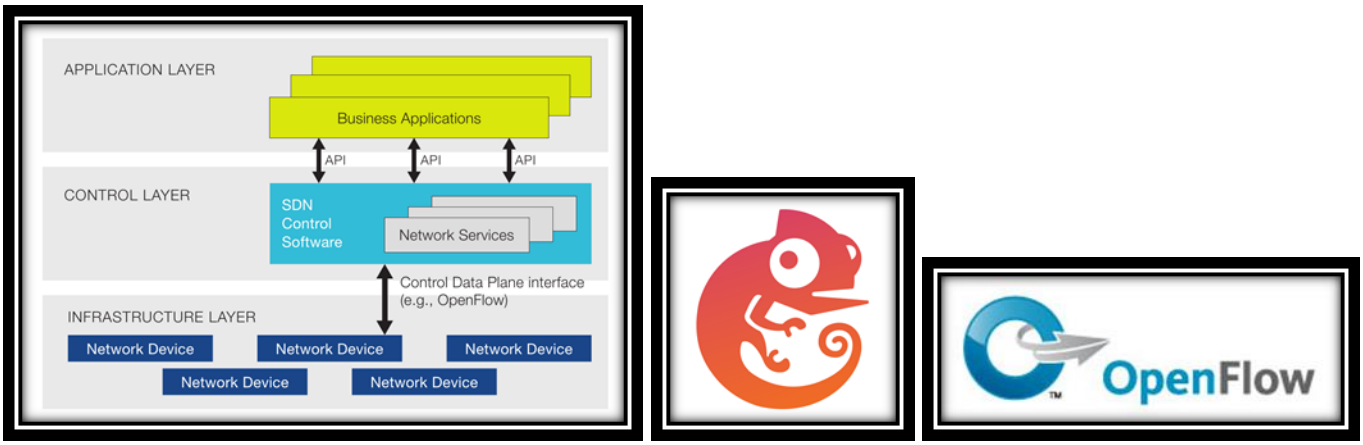
Determininstic Data ={ port 6633,interfaces }
Non Determininstic Data = {DHCP IP address allocation}

e = end state {Devices Powered Down/Termination of Network connectivity}

1. DESIGN AND ANALYSIS OF SYSTEM

The implemented topology of Software Defined Network in GNS3 comprises of following:

- ***Installing Open Daylight Controller on Docker Container***
- ***Integrate Open Daylight with OpenFlow Switches (OVS) in Docker Container.***



Compilation of SDN as framework + GNS3 as a tool+ Openflow as communication protocol

➢ About GNS3

GNS3 is a free graphical network simulator capable of emulating a number of network devices. This makes it possible for anyone to quickly and easily spin up network hardware for testing and educational purposes without the heavy expense of physical hardware. Supported devices include Cisco routers and firewalls, Juniper routers, and frame-relay switches.

➢ Open Daylight Controller

The Open Daylight platform (ODL) provides a flexible common platform underpinning a wide breadth of applications and Use Cases.

Use Cases for ODL: -

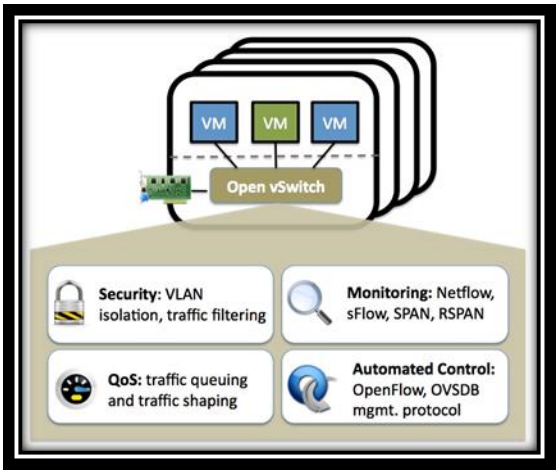
Automated Service Delivery
Cloud and NFV
Network Resource Optimization
Network Visibility and Control

➢ Docker Containers support in GNS3

The Docker support is a new feature of GNS3 1.5. This features has been started by Goran Cetusic during the Google Summer Of Code and finished by the GNS3 core team.
Containers use the host kernel this mean they consume, less less RAM and CPU. Docker containers are available from a registry, you can fork them in order to add your own tools.

- **Open vSwitch Switches: -**

Open vSwitch is a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license. It is designed to enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols (e.g. NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, 802.1ag). In addition, it is designed to support distribution across multiple physical servers similar to VMware's vNetwork distributed vswitch or Cisco's Nexus 1000V.



- **Internet for GNS3 VM: -**

This appliance simulates a domestic modem. It provides an IP via DHCP and will nat all connection to the internet without the need of using a cloud interface in your topologies. IP will be in the subnet 172.16.0.0/16. Multiple internet will have different IP range from 172.16.1.0/24 to 172.16.253.0/24.

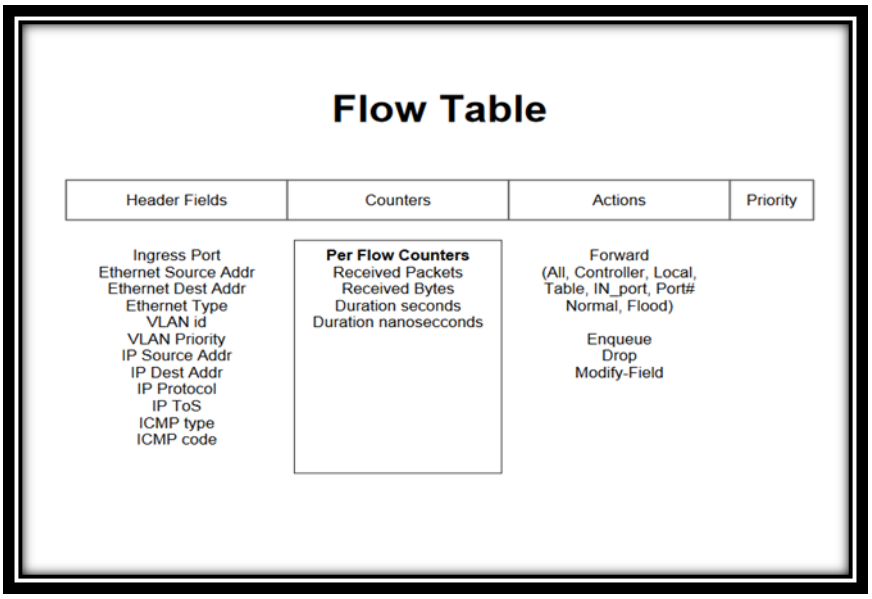
- **Open Flow TABLE: -**

A flow is defined as all the packets matching a flow-entry in a switch's flow-table. Flow entries are quite general, and resemble ACL entries found in firewalls—although here they can include fields from layers 2, 3 and 4.

As the links in the question suggest, flow-table is a broad term. A forwarding table uses IP address or MAC addresses to decide the next hop for the packet.

But a flow table may use any of the information within the packet to decide the next hop for it e.g. incoming switch port, TCP port/IP address (src or dst), VLAN tag etc.

Given below is an example of Flow Table parameters list.



- **Firefox 31.1.1~2-1: -**

A GNS3 Docker container and a light Linux based on TinyCore Linux with Firefox preinstalled.
It runs as a Firefox VM to provide Interface to the Open Daylight Controller that runs on the Ubuntu Server

- **OpenFlow PROTOCOL**

OpenFlow is an open standard that enables researchers to run experimental protocols in the campus networks we use every day. OpenFlow is added as a feature to commercial Ethernet switches, routers and wireless access points – and provides a standardized hook to allow researchers to run experiments, without requiring vendors to expose the internal workings of their network devices. OpenFlow is currently being implemented by major vendors, with OpenFlow-enabled switches now commercially available.

- **OpenFlow Protocol Working**

In a classical router or switch, the fast packet forwarding (data path) and the high-level routing decisions (control path) occur on the same device. An OpenFlow Switch separates these two functions. The data path portion still resides on the switch, while high-level routing decisions are moved to a separate controller, typically a standard server. The OpenFlow Switch and Controller communicate via the OpenFlow protocol, which defines messages, such as packet-received, send-packet-out, modify-forwarding-table, and get-stats.

The data path of an OpenFlow Switch presents a clean flow table abstraction; each flow table entry contains a set of packet fields to match, and an action (such as send-out-port, modify-field, or drop). When an

OpenFlow Switch receives a packet it has never seen before, for which it has no matching flow entries, it sends this packet to the controller. The controller then makes a decision on how to handle this packet. It can drop the packet, or it can add a flow entry directing the switch on how to forward similar packets in the future.

- **OpenFlow 1.3 Features: -**

IPv6 extension headers: Can check if Hop-by-hop, Router, Fragmentation, Destination options, Authentication, Encrypted Security Payload (ESP), unknown extension headers are present

*MPLS Bottom-of-Stack bit matching

*MAC-in-MAC encapsulation

*Tunnel ID meta data: Support for tunnels (VxLAN, …)

*Per-Connection Event Filtering: Better filtering of connections to multiple controllers

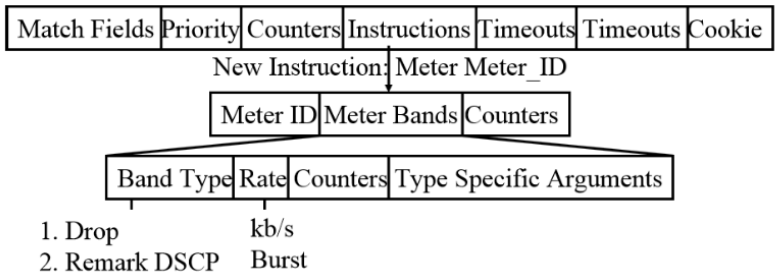
*Many auxiliary connections to the controller allow to exploit parallelism

*Better capability negotiation: Requests can span multiple messages

*More general experimenter capabilities allowed

A separate flow entry for table miss action.

- **OpenFlow v1.3 Header Format**



2. DISCUSSION ON IMPLEMENTATION RESULTS

The following is a set up of an Software Defined Network topolgy that runs virtually within Docker Container in GNS3. All the appliances viz Ubuntu Server,Firefox Browser Terminal,OpenVSwitch switches are self contained within docker container of GNS3.

According to the architecture on the top that is the OpenDaylight Controller used for monitoring the network flow. Address allocation of the Ubuntu Server,OpenVSwitch is done dynamically by the Internet Cloud.

Step for configuring appliance to obtain dynamic address: -

- Halt all the running devices to be configured.
- Right click on the appliance and choose the "Configure" option/ for ver 2.0.0 above select Edit Config.
- For older versions in the Config pane select "Edit" and uncomment the 2 lines below

  "#DHCP Config for eth0"
- Save the configurations. And check the allocated IP using the "ipconfig" command.

Requirements for Ubuntu Server, starting ODL controller and Installing features in ODL:

- Pre-Configured Java Environment
  - **Commands to run on Ubuntu Server**-

    *apt-get update
    *apt-get install default-jre-headles
    *export JAVA_HOME=/usr/lib/jvm/default-java
- Install the wget and unzip utility to download and extract the ODL packages from internet.
- Navigate to the path of the ODL directory (cd distribution-karaf-0.5.2-Boron-SR2/)
- Execute the ./karaf file and start the ODL Server.
- To install features in ODL type the following command in the ODL terminal

(feature:install odl-restconf odl-l2switch-switch odl-mdsal-apidocs odl-dlux-all)

Configuration on the OpenVSwitch Management Devices: -

- **ovs-ofctl**: OpenFlow switch management utility

- ovs-ofctl -O OpenFlow13 dump-flows br0 (Shows the flows being written to to OpenFlow switches)

- **ovs-vsctl**: OpenFlow ovs-vswitchd management utility

  - The ovs-vsctl is used primarily to manage the ovs-vswitchd. The ovs-vswitchd is connected with ovs-server which manages it to save and change the configuration into a database.

1. ovs-vsctl set bridge br0 stp_enable=true (Enable Spanning Tree Protocol to remove the effect of loop within OpenVSwitches)

2. ovs-vsctl set-controller br0 tcp:UbuntuServerIP:6633 (Configure switches ton talk to ODL server)

3. ovs-vsctl show (Shows switch is connected to the controller)

- **Algorithm**

  - Set up the required topology by dragging the appliances and devices to the workspace.

  - Establish connection amongst the devices through cable connectors.

  - Start up the devices.

  - Obtain IP address for Ubuntu server,OpenVSwitch switches.

  - Check connectivity using “ping” utlility with the ubuntu server from Firefox terminal & OpenVSwitches.

  - Start the Open Daylight Controller from the Ubuntu server.

  - Connect to the Open Daylight dlux from Firefox terminal.

  - Configure switches to talk to ODL server using ovs-vsctl utility/command mentioned above.

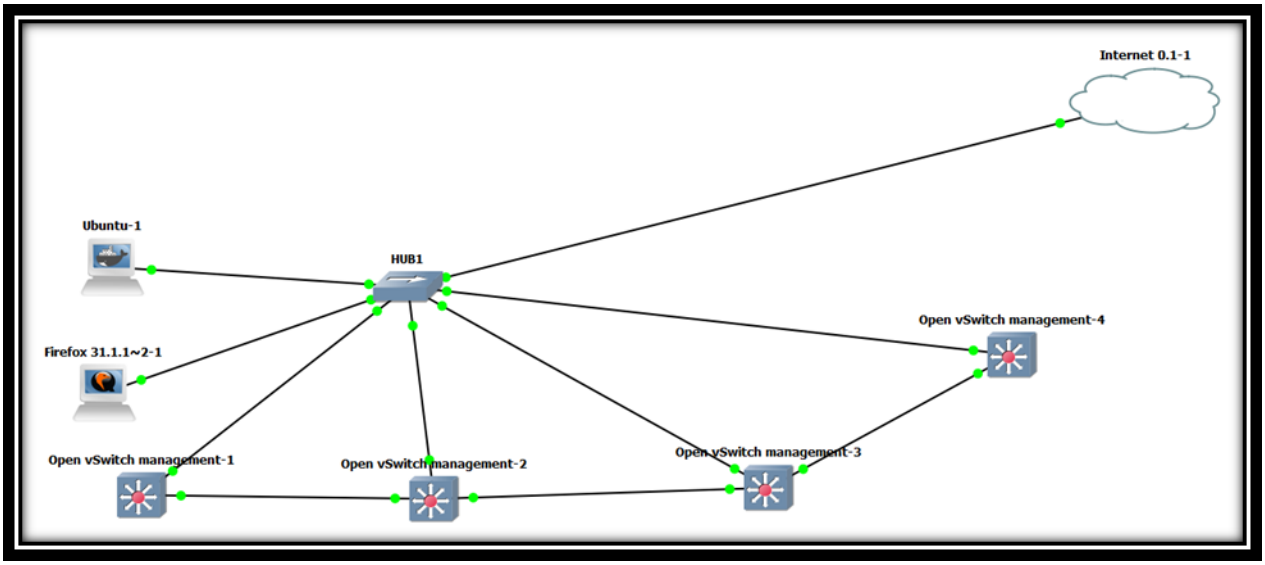
  - Login the ODL interface using credentials (Username-admin Password-admin)

  - Reload the Interface on Firefox browser and obbser the Topolgo,Table Details.

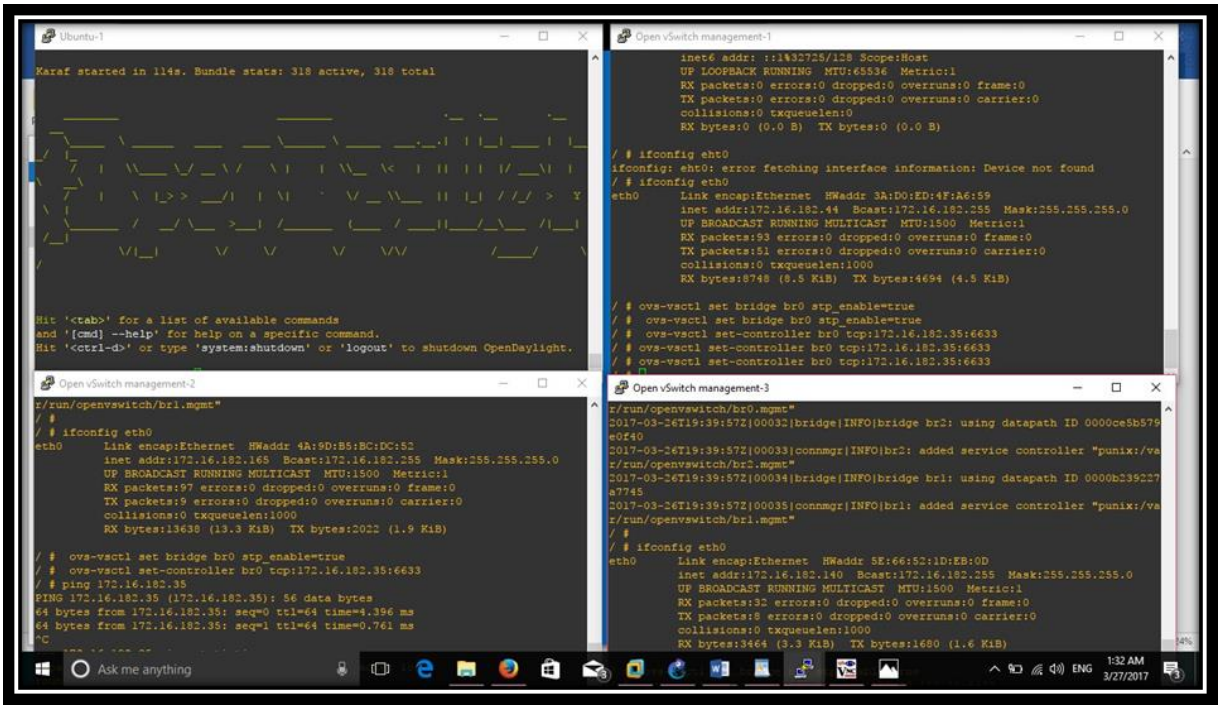
  - Using ping or any desired utlility to check connectivity amongst the switches.

Implementation Snapshots: -

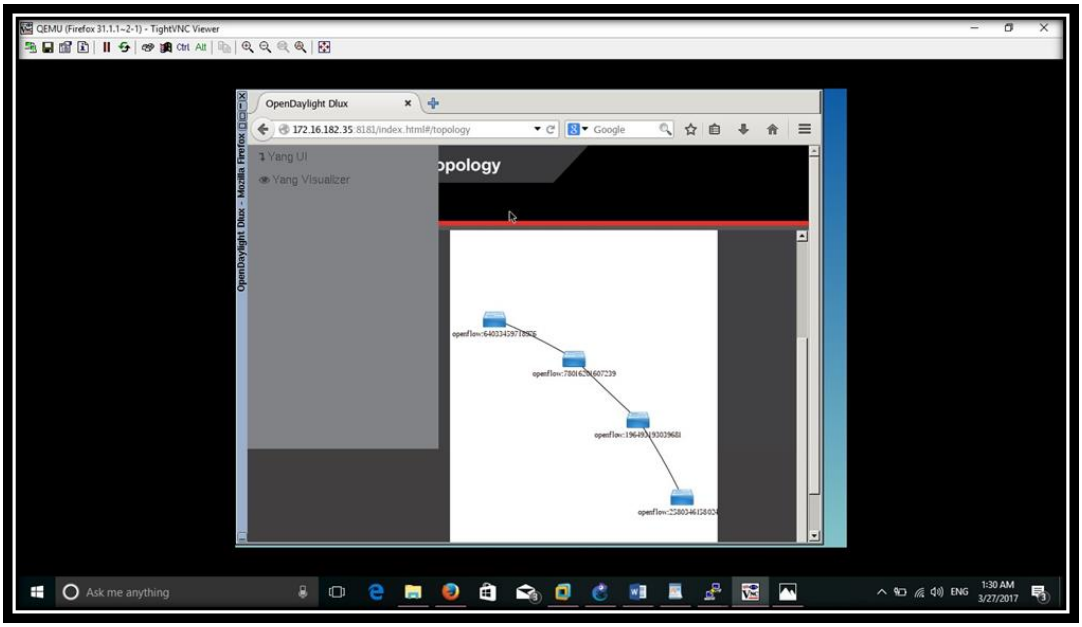
- ***Actual designed topology in Graphical Network Simulator-3***



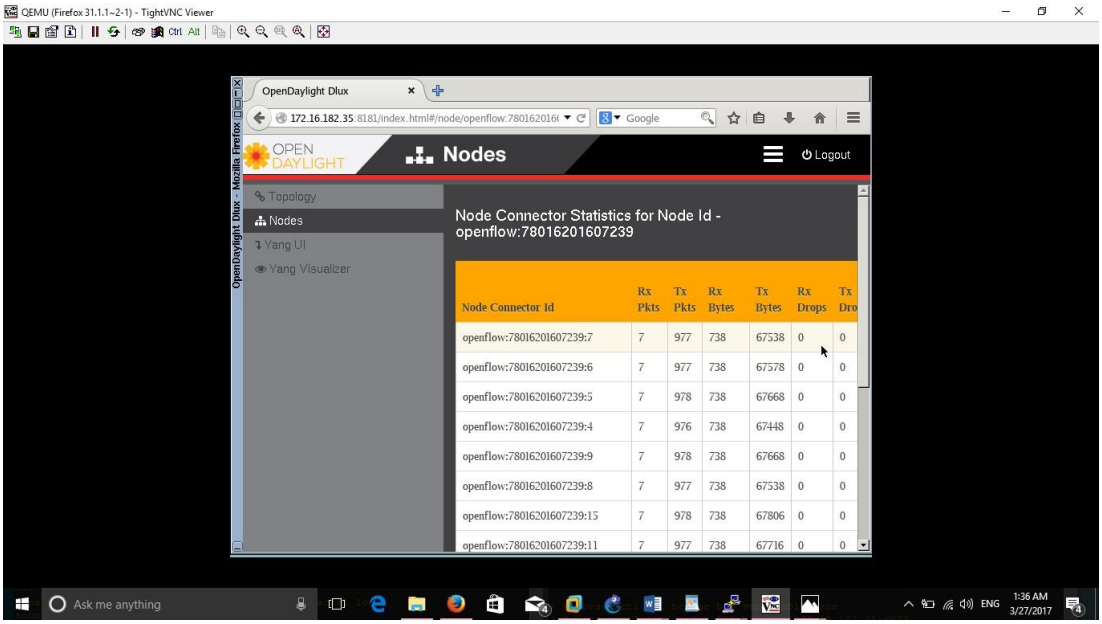
- ***Terminal Configuration of Ubuntu Server and Ovs***



- ***ODL Firefox Browser UI displaying connected Open vSwitch switches***



- ***Flows being written on the Open vSwitch switches***



3. CONCLUSION AND FUTURE ENHANCEMENT

In a nutshell, we were presented a future networking concept of Software-Defined Network along with its terminologies those were the OpenFlow Protocols, Devices like Open vSwitch switches, Ubuntu server,Firefox VM (running in a Docker Container) within Graphical Network Simulator-3

Also supported with a small demonstration of an Open Flow network that runs virtually within GNS3

- ***Future enhancement***

Various networking vendors and vendor-consortiums are trying to address these goals in different ways. Some strategies are:

- Creating network controllers (Big Switch's focus)
- Defining standard protocols between future network controllers and data switches (Openflow and others)
- Virtualizing network routers, firewalls, and load balancers (Nicira's focus)
- Creating a vendor independent "Network OS" (Open Daylight's mission)
- Creating proprietary top-to-bottom network stacks (Cisco)

4. REFERENCES

[1] April 13, 2012 Software-Defined Networking: The New Norm for Networks ONF White Paper Open Networking Foundation, Palo Alto, CA 94303

[2] December 30,Nick Feamster, Jennifer Rexford and Ellen Zegura. The Road to SDN: An Intellectual History of Programmable Networks. ACM Queue

[3] Business white paper | Build the foundation for SDN with OpenFlow, HP Foundation.

[4]GNS3 ACADEMY , GNS3 TALKS: OPENDAYLIGHT INSTALLATION: UBUNTU DOCKER CONTAINER, OPENDAYLIGHT, PYTHON, SDN