ISSN No. 0976-5697

Volume 8, No. 5, May-June 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Open Daylight as a Controller for Software Defined Networking

Sumit Badotra
Department of Computer Science and Engineering
Shaheed Bhagat Singh State Technical Campus, Ferozepur, Punjab, India

Japinder Singh
Department of Computer Science and Engineering
Shaheed Bhagat Singh State Technical Campus, Ferozepur, Punjab, India

Abstract: For making programmable networks which are responsive and flexible to organizations as well as to users, an industry movement has started called as Software-Defined Networking (SDN). Open Daylight the comprehensive open source controller of SDN, which is helping to make it possible, around a common SDN platform by combining the industry as well the Open Daylight community it helps not only to solution providers, developers but also to all the users who are working together for delivering interoperable, programmable networks to all service providers, various enterprises, universities and a number of organizations around the world. In this paper an overview on Open Daylight is provided with its history, architecture, various released versions, installation steps followed by its features.

Keywords: Software Defined Networking, Open Daylight NETCONF, YANG, Service Abstraction Layer Open Flow Network Function Virtualization, Open Network Foundation.

1. INTRODUCTION

Due to having the immense amount of data nowadays, this ultimately resulted into large data centers. Further due to the availability of large compute and storage through these large data centers it resulted into new business models. Increase in the complexity of computing and storing this data has also resulted into more complex computer networks for a network administrator [1]. In a data centre, with virtual machines virtual networks are now common along with. Managing networking in data centers is a difficult task which requires innovation through transformation and there the new technology has emerged called as Software Defined Networking (SDN) [1]. Today computer networks are very complex as more and more devices are increasing day by day along with the content they access. The kind of equipment used in networks like Intrusion Detection system,switches,firewalls,Load balancers are typically very hard to manage by network administrator individually, the solution for this is Software Defined Networking. It has changed the way we used to manage the networks. The two main basic principles of Software Defined Networking (SDN) are 1) It separates the control plane from data plane (control plane contains the intelligence, control logic while data plane contains the physical infrastructure or low level network elements that are used for packet forwarding and switching) 2) Control plane acts as a brain of the network which has a direct control over the Data plane, all the elements in the Data plane can be manipulated as per the needs, there is no need to configure each and every element of data plane individually. Network Function Virtualization (NFV) and Software Defined Networking (SDN) complement each other, although they do not depend upon each other [2].

For designing, managing and decoupling networking services Network Function Virtualization (NFV) is used. The network functions such as Domain Name Services (DNS), Network Address Translation (NAT), and Intrusion Detection System (IDS) etc. are decoupled by the NFV from propriety hardware appliances as they can run in software [2].For providing the multiple services to the customers, service provider has to implement the multiple virtual network function (VNF) instead of single virtual network function (VNF) ,the new concept came where multiple virtual Network Function(VNF) are providing services through concatenation is called as service chaining after this Quality of Service (QOS) is the issue to handle in for [2]. NFV standardization committee is ETSI NFV group. On the other hand SDN is an emerging technology which separates central logic from its infrastructure. SDN can be centrally managed and dynamic changes can be made, it is cost effective and easy to use. It is founded by Open Networking Foundation (ONF) group.

2. OPENDAYLIGHT OVERVIEW

Open source platform for Software Defined Networking (SDN) is OpenDaylight .To provide the centralized, programmatic control as well as network device monitoring open protocols are used. Like many other SDN controllers, such as Ryu, Pox etc.OpenDaylight also supports OpenFlow, as well as offering ready-to-install network solutions as part of its platform [3]. An interface for the devices that comprise your computer is provided by your operating system in the same way, to connect network devices quickly and intelligently for optimal network performance an interface is also provided by the OpenDaylight.Fig.1 shows the logo of the OpenDaylight.

Fig.1 OpenDaylight logo

Hosted by the Linux Foundation OpenDaylight is a collaborative open-source project. To accelerate the adoption of SDN and creating a solid foundation for NFV as well is the primary goal of this project [3]. A guarantee for

an open, community decision making process on business and technical issues is made and for achieving the goal with the project of OpenDaylight, a community has come together through the unity of open community developers, open-source code and project governance as well.

For any SDN architecture OpenDaylight can be a core component. Due to the open-source nature of the controller it enables the users to minimize operational complexity, and hence extending the life of their existing infrastructure. Basically hardware and enable new services, capabilities are only available with SDN [3]. For different kind of enterprises such as enterprise IT providers, network service providers or cloud services provider such open-source controller framework can be of great utility.

As shown in fig.2 the architecture of OpenDaylight it is multi layered architecture; the main layer is controller platform because controller resides in it, and acts as a brain to the network because it manages the flow of traffic from switches using flow tables [3,4].The OpenDaylight Controller acts as a pure software and it can be run on any Operating System as a JVM (java virtual machine) and Metal as long as it supports Java. Multiple protocols (as plug-in), e.g. OpenFlow 1.0, OpenFlow 1.3, BGP-LS, etc on the Southbound can be supported.

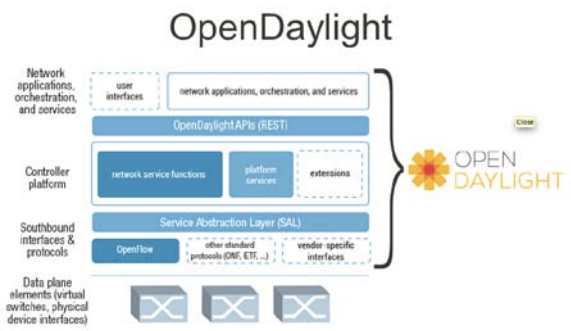

Fig.2 OpenDaylight Architecture

Service Abstraction Layer is located at the heart of the modular design of the Controller as shown in the fig.2 of OpenDaylight architecture and to support multiple protocols on the Southbound and providing consistent services for modules and Apps (where the business logic is embedded) is allowed by it only.

The starting of Open Daylight Controller is initiated with an OpenFlow 1.0 Southbound plug in. As part of their contributions/projects etc. other Open Daylight contributors would add to those. Into a Service Abstraction Layer (SAL) the linking of these modules is done dynamically, In between the Controller and the network devices to fulfill the requested service irrespective of the underlying protocol used is figured out by the SAL [4].

There are some dynamically pluggable modules, present which are responsible for performing network tasks and are contained in the controller itself in OpenDaylight. To insert other services and extensions for enhanced SDN functionality it is also possible in it. These all modules are linked to a Service Abstraction Layer (SAL) dynamically.

To fulfill the requested services independently of the underlying protocol used and the network devices the infrastructure layer is exposed by the SAL to the applications north of it. When programming applications for OpenDaylight is concerned there are two different approaches to the SAL that can be taken into account [4]:

- The API-Driven SAL (AD-SAL)
- And the Model-Driven SAL (MD-SAL).

AD-SAL

The AD-SAL approach has the following main characteristics:

- It can be used with both southbound and northbound plug gins.
- It is stateless.
- It is limited to flow-capable devices and services only [4,5].
- The applications are programmed into the controller as OSGi bundles.
- The flow programming is reactive, by receiving events from the network

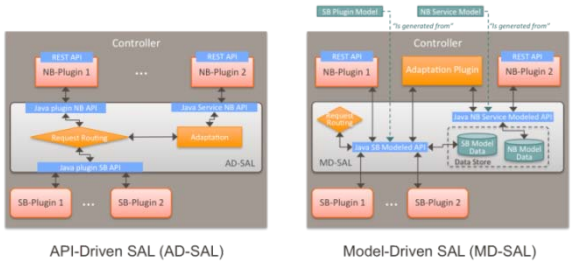

Fig.3 architectural diagram of API-Driven SAL and Model-Driven SAL

MD-SAL

This approach has the following features:

- It has a common REST API for all the modules.
- It can store data for models in permanent or volatile APIs.
- It is model agnostic. It supports any device or service models.
- The applications are programmed outside the controller.
- The flow programming is proactive, without the possibility to receive events from the network [5].

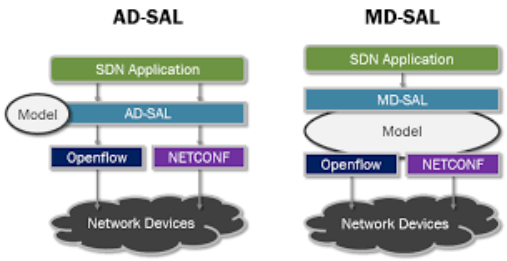

Fig.4 working model of AD-SAL and MD-SAL

OpenDaylight has the structure of a SDN environment. The controller exposes open northbound APIs which are used by applications. It supports the OSGi framework and bidirectional REST for the northbound API.OSGi is a modular system and service platform for the Java programming language that implements a completely dynamic component model, something that does not exist in standalone JVM environments.

While REST is used by applications running outside the controller itself, and even in different machines, the inner applications use OSGi [5]. All in all, the applications are the ones providing the logic, and using the controller together

network intelligence or run algorithms, and then to orchestrate the new rules throughout the network.

3. HISTORY

On February 8, 2013, a new coalition forming around SDN was announced by the SDN Central. One month later, on April 8, [6] announcement of founding the OpenDaylight Project is done by the Linux Foundation, as a community-led and industry-supported framework to promote the adoption of the new network paradigms of SDN and NFV.The original founders of the project were: Arista Networks, Big Switch Networks, Brocade, Cisco, Citrix, Ericsson, HP, IBM, Juniper Networks, Microsoft, NEC, Nuage Networks, PLUMgrid, Red Hat and VMware. They committed to providing economical and engineering resources to help in the development of the platform [6].

3.1 Releases:

Since the creation of the project, there have been these major releases of the OpenDaylight controller [7]:

- **Hydrogen** (February2014)
- **Helium** (September2014)
- **Lithium** (August2015)
- **Beryllium** (March2016)
- **Boron** (December 2016)
- **Carbon** (Current Release)

3.1.1 Hydrogen:

It was released on February 4, 2014. In three different editions it was delivered, each one of the edition is oriented to a different kind of user: The three editions of it are as follows:

• ***Base Edition***: It is meant for those who are exploring SDN and academic initiatives or OpenFlow in physical or virtual environments this edition is oriented towards them.

• ***Virtualization Edition***: This edition is designed for data centers and also included the basics plus functionality for creating VTN (Virtual Tenant Networks) and virtual Overlays, as well as applications for security and network management [7].

• ***Service Provider Edition***: Oriented towards providers and carriers who are managing the existing networks and wanted to start using SDN and NFV. Protocol support as well as security and network management applications are also included in this service provider edition. The architectural diagram of the Hydrogen is provided in fig.5.

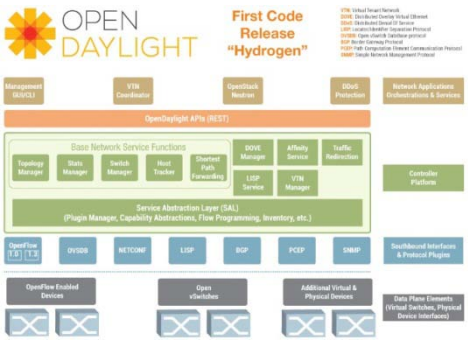

Fig.5 Architecture of Hydrogen released version of OpenDaylight

3.1.2 Helium:

Firstly it was released on September 29, 2014. However, some of its revisions of it were being released until March 2015. It got the idea of different editions and was released as a unique version as compared to other versions of OpenDaylight. Introduction of karaf as the tool to manage the controller is done in this version only [7]. The dynamic management of the available modules, so that, starting from the base controller; the user can install the features accordingly which satisfy its particular needs are included in it only. The architectural diagram of the Helium is provided in fig.5.1

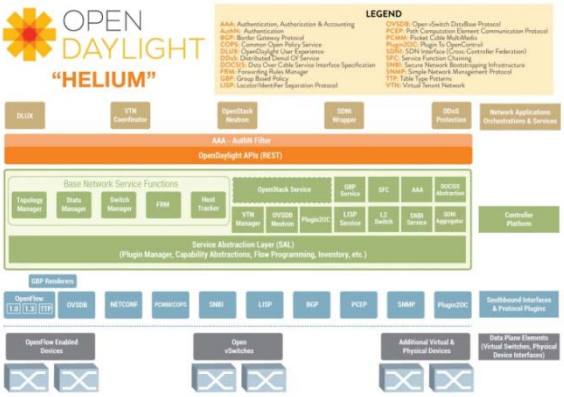

Fig.5.1 Architecture of Helium released version of OpenDaylight

3.1.3 Lithium:

It was released on June 29, 2015 and the basic idea it followed was introduced by Helium itself. On broadening the programmability of intelligent networks a particular focus is made. It also introduced some new features in many of the functionalities included in the OpenDaylight controller [7].

For an OpenDaylight, which is an open source platform for building programmable, software-defined networks Lithium is the third release. With the help of this released version of lithium now more number of service providers and enterprises can transition to SDN with particular focus only on broadening the programmability of making the networks intelligent [7].They can compose their own service architectures or leverage an OpenDaylight-based commercial offering to deliver dynamic network services in a cloud environment, craft dynamic intent-based policies and begin virtualzing functions with Service Function Chaining (SFC).The architectural diagram of lithium is shown in fig.5.2.

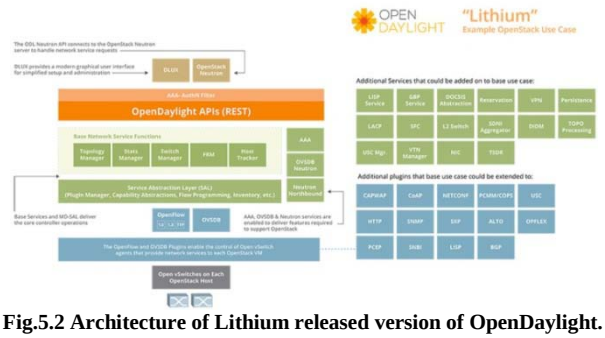

Fig.5.2 Architecture of Lithium released version of OpenDaylight.

3.1.4 Beryllium:

Beryllium (Be) is the fourth release of OpenDaylight (ODL) that leads the open source platform for programmable and software-defined networks. ODL is the industry's SDN platform, for supporting a broad set of use cases and thus provisioning the foundation for networks of the future.

To solve many key network challenges related to Network Resource Optimization, Cloud and NFV; Research, Education and Government etc. all enterprises are using OpenDaylight [8].

To strength the architecture of ODL it uses the Model which is based on Driven Service Abstraction Layer (MD-SAL) and delivery of high scale and the ability to easily incorporate new applications as well as protocols is easily achieved [8]. The architecture of Boron is a shown in the figure 5.3

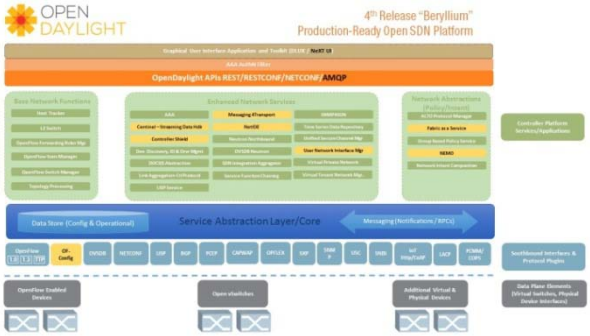

Fig.5.3 Architecture of Beryllium released version of OpenDaylight.

3.1.5 Boron:

Boron (B) is the fifth release of OpenDaylight (ODL), which is the leading open source platform for all programmable as well as software-defined networks [9]. OpenDaylight has become the platform for service providers and enterprises making the amendments to the networks.

With the release of Boron, a new mark towards the path of OpenDaylight in the field of technology and community maturity is achieved. Boron is the result of significant collaboration between users, network equipment vendors [9]. For building OpenDaylight-based solutions to make use of unique use cases and user requirements is done by the growing ecosystem of systems integrators and application developers .By using their own all deployment experiments and experiences, due to this many of the leading user organizations have already invested their own resources into the OpenDaylight developer community. (More than half of the new projects are proposed only for Boron).

A strong practical focus on two leading types of deployments is provided by the Boron which gives all the enhancements to cloud and NFV support as well as large-scale network engineering [9]. The improvement of performance and documentation is enhanced by the new operational tooling. Delivering the new tooling and documentation to support application developers is also provided by the boron. The greater integration with larger industry frame works from OPNFV and OpenStack to CORD and Atrium Enterprise is also done [9].

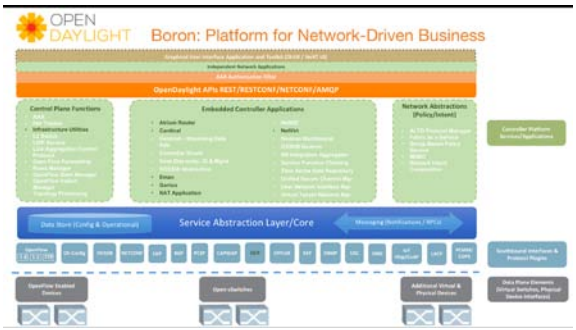

Fig.5.4 Architecture of Boron released version of OpenDaylight.

3.1.6 Carbon:

This is the current release plan of OpenDaylight and it is the fifth release of OpenDaylight and this release plan has some differences from its predecessors [10]. The difference in its features from already released versions of OpenDaylight is as follows:

- **Feature:** Grouping of code and functionality in a project in a logical way [10]. While this Feature is usually a Karaf feature but it could also be any other component or grouping for better performance.
- **Top-Level Feature:** One of the major pieces of functionality delivered by a project is provided this Feature. There is no requirement of understanding to know the internals that how and when to install it as such. Most of the projects will have a small number of; Top-level features maybe even only one. But in many cases this could be the only meta-feature grouping together lower-level features [10].
- **User-Facing Feature:** when somebody looking to install and run OpenDaylight it is a Top-Level Feature that should be known [10]. Installation of it should be accomplished by them and be able to tell that it's been installed in the form of new user interface elements, which ultimately support for new southbound devices, or other mechanisms [10].

Release Distributions of carbon:

- **Stable Distribution of carbon:** A Karaf stable distribution containing of carbon contains the collection of all the Stable Features as when the compilation of Carbon Stable Release Feature is done all repositories hosted in the Integration project [10].
- **Extended Distribution of carbon:** A Karaf extended distribution contains the collection of both Stable and Extended Features because repository hosted in the Integration project when they are compiled in the Carbon Extended Release Feature [10].

4. INSTALLATION PROCEDURE

Step 1: Install any virtual machine (like VMware or virtual box).

Step 2: Make a separate machine of Ubuntu by the name of OpenDaylight and download the latest released version of OpenDaylight in it. (In our case we have downloaded Ubuntu 14.04 and beryllium released version of OpenDaylight).Download it from www.opendaylight.org/downloads.

Step3: Make two separate machines of Ubuntu and mininet on virtual box (you can download the mininet from www.mininet.org) as shown in the fig.4

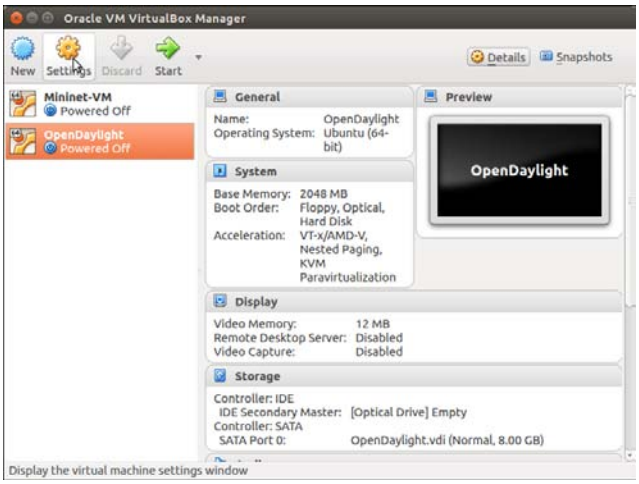

Fig.6 creating two machines on virtual box

Step 4: The OpenDaylight SDN controller is a Java program so install the Java run-time environment with the following command:

- *-sudo apt-get install openjdk-7-jdk*
- *-sudo apt-get install openjdk-7-jre*

Step 5: Install maven (The project management and comprehension tool is called as Maven a complete build lifecycle of framework is provided by it. Automation of the project's build infrastructure by the development team is done in almost no time because layout of standard directory and a default build lifecycle is used by the Maven [11].

Maven can set-up the way to work as per standards in a very short time in case of multiple development teams environment Because most of the project setups are simple and reusable, it makes life of developer easy as it creates the reports and checks the building and testing automation setups also [11].

- *-sudo apt-get install maven*

Step 6: Extract the zip file of the released version of OpenDaylight which you done in step 2. (We gave the names as odl.zip and odl to the extracted files respectively) as shown in the figure 4.1.

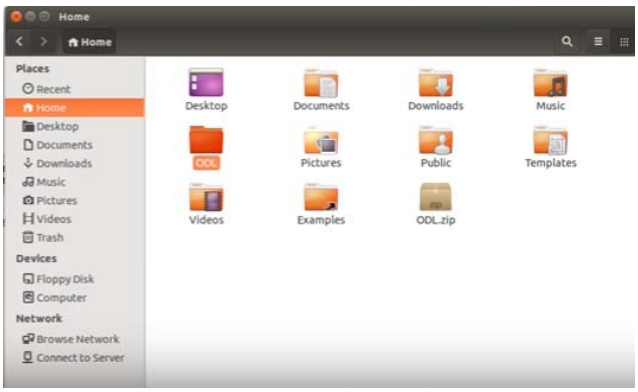

Fig 6.1 making two separate folders after extraction.

Now run the following command on Ubuntu terminal no.1

- *cd ODL/bin*
- *:-/ODL/bin$ ./karaf –of13*

After running above stated commands following screen will appear as shown in the figure 4.2. It will accelerate the karaf feature [12]. The deployment of OSGi applications is supported by the apache karaf which is an OSGi container. In OSGi, dependency of one bundle on another is there. So, it implies the fact that most of the time, you have to first deploy a lot of other bundles (due to the dependency upon each other) required by the application to deploy an OSGi application, it increases the overhead also.

Providing a simple and flexible way to provision applications is the basic feature of apache karaf. The application provisioning is an Apache Karaf feature in apache karaf [12].

Fig.6.2 accelerating the karaf feature

Step 7: Open the terminal no.2 in Ubuntu and run the following command

- *Sudo apt-get-install nmap*
- *nmap local host*

Nmap (*Network Mapper*) originally written by Gordon Lyon is a security scanner and it is used for discovering the hosts and services on a computer network and then finally building a "map" of the network that is why called as nmap. To achieve its goal, sending specially crafted packets by the nmap to the target host(s) is done and then analyzes the responses provided by them [13].

Step 7: Install the features to the odl in the terminal no.1 where karaf features were installed using the following commands:

>feature

>feature: install odl-l2switch-switch-ui

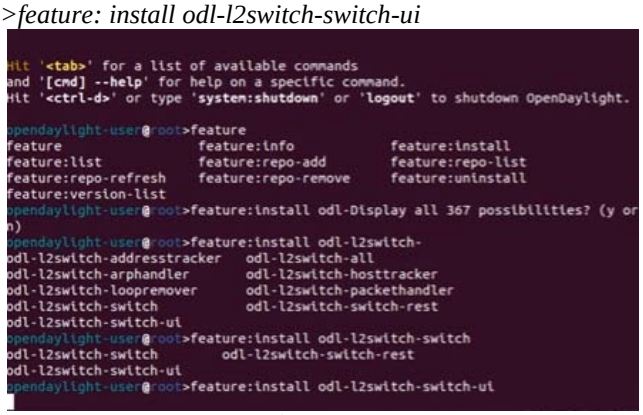

Fig.6.3 adding features to the controller

>feature: install odl-dlux-all (OpenDaylight doesn't ship with any features installed by default. You'll need to use the Karaf console to install the features required by DLUX) [14].

In the terminal no.2 run the command *ifconfig*

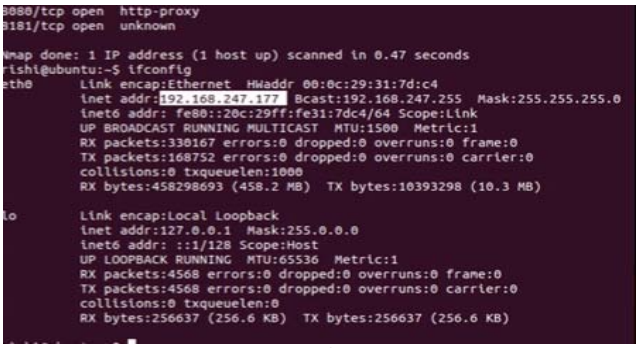

Fig 6.4 IP address of mininet machine

And go to the URL as *192.168.247.177:8181/index.html* as shown in the figure

Fig.6.5 web interface in OpenDaylight

and then use

- *Username- admin*
- *Password- admin.*

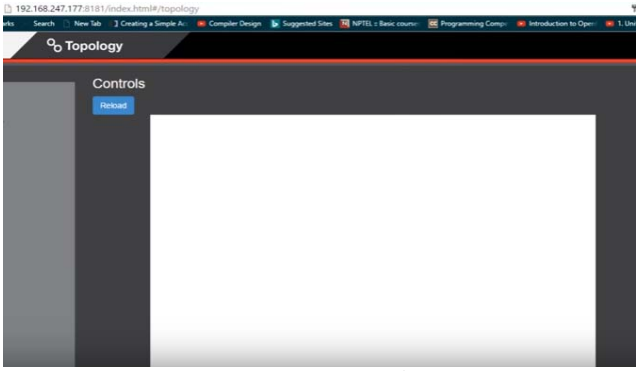

Fig.6.6 cont.web interface

Step 8: Open the second machine (Mininet) on your virtual box.
Put the username-mininet
Password-mininet

Fig.6.7 turning on the mininet machine

Now on the terminal no.2 of Ubuntu machine run the following command

ssh-X mininet@192.168.247.158 (this is our ip of mininet machine and we have obtained it using the command *ifconfig*)
Now,we are remotely accessing the mininet on the Ubuntu machine,we will create a topology in which there are 16 hosts and 15 switches by using the command as follows:
sudo mn –controller=remote, ip=192.168.247.177--topo=tree,4,2

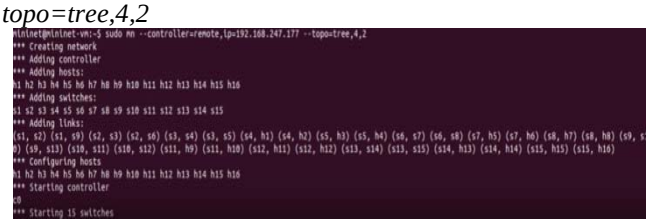

Fig.6.8 creating the topology

This will create a topology of tree with corresponding number of switches and hosts. Now to have a look (graphically) on these topologies go to the web address *192.168.247.177:8181/index.html#/topology*

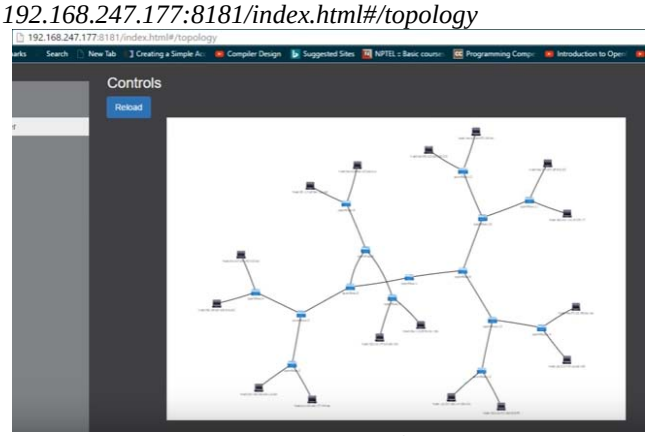

Fig.6.9 graphical representation of the tree topology

With the help of Yang visualizer we can change the statictics-work-node ,get flow statistics and many other operations.

Fig.6.10 different operations of yang visualize.

Fig.6.11 get flow statistics

7. CONCLUSION

OpenDaylight as an SDN controller offers many possibilities as we have already seen this in our document .The controller is not at all difficult to manage, although all experiments have been done on mininet because in order to work, already existing tools such as Maven or OSGI are used. The two approaches are used to program applications, AD-SAL and MD-SAL. Depending upon the choice these two help to choose between two approaches called as reactive approach or proactive approach. OpenDaylight is supposed to be one of the most documented controllers. All in all, with a massive industry support and a constant updating allowing compatibility of OpenDaylight with more protocols and standards, and hence in future OpenDaylight is very likely going to become an essential part of the developing of telecommunication networks.

8. ACKNOWLEDGEMENT

We would like to thank almighty for his constant blessings. Then we would like to dedicate our gratitude towards parents, family, friends, and in essence, all sentient one beings.

9. REFERENCES

[1] Khattak, Muhammad Awais and Adnan Iqbal "Performance evaluation of OpenDaylight SDN controller" Parallel and Distributed Systems (ICPADS), 20th IEEE International Conference 16-19 December, 2014.

[2] Jammal, Manar, Taranpreet Singh, Abdallah Shami , Rasool Asal , Yiming Li"Software defined networking: State of the art and research challenges" *Elsevier computer Networks* 72(2014)74-98.

[3] "Software-Defined Networking: The New Norm for Networks," ONF White Paper, April 13, 2012

[4] Medved,Jan, Anton Tkacik, Robert Varga, Ken Gray"OpenDaylight: Towards a Model-Driven SDN Controller Architecture".

[5] "OSGi", http://www.osgi.org/

[6] "OpenDaylight Project," https://wiki.opendaylight.org/view/Main_Page

[7] "OpenDaylight" https://www.opendaylight.org/

[8] "OpenDaylight Documentation" Release Beryllium OpenDaylight Project Sep 06, 2016.

[9] "Release boron at" https://www.opendaylight.org/odlboron

[10] "Release carbon at" https://wiki.opendaylight.org/view/Simultaneous_Release:Carbon_Release_Plan

[11] "Maven at" https://maven.apache.org/

[12] " Apache karaf feature documentation at" http://karaf.apache.org/documentation.html

[13] "Nmap (network Mapper)" at https://nmap.org/docs.html

[14] "Documentation of karaf feature at" https://karaf.apache.org/documentation.html