International Journal for Research in Engineering Application & Management (IJREAM)
ISSN : 2454-9150 Vol-06, Issue-09, DEC 2020

Constructing an Effective and Dedicated SDN Using Floodlight Controller

*Twasul Mukhtar Mahmoud, #Raeid Ibrahim Abbas

*,#Department of Electronics Engineering, Faculty of Engineering, University of Garden City, Khartoum, Sudan. *tawasul.mukhtar@gmail.com, #raiedin2018@gmail.com

Abstract - SDN technology is an approach to network management that enables dynamic, programmatically efficient network configuration in order to improve network performance and monitoring, making it more like cloud computing.[1].The advanced in software defined network (SDN) technology around the globe design and management added many merits and characteristics that facilitated complex operations on traditional network , the dependence only on the controller to make the decision over the network, Which was performed through the controller to solve all operations carried out by several devices to aggregate the functions of the firewall, router, switch, and servers devices which are combined into software programs with different modules to be activated as needed inside the controller.

In this paper we designed the network simulation with floodlight controller and applied different applications in Network traffic Restricting and allocating which makes data transfer more professional and effective by controlling Restricting and allocating data, which makes data transfer more professional and effective, by controlling and assigning addresses the ports , IP addresses and Mac addresses of sources and destinations.

Key words: Software Defined Networks, controller, mininet Emulator, floodlight .

I. INTRODUCTION

Software-defined networking (SDN) is a new emerging technology in the field of computer networks which is evolved from the work done at Stanford University and Berkeley University around 2008. It allows network administrator to manage the network services through the abstraction of lower level functionality, which is done by separating the control layer (brain of network) from the data layer (forwarding the packets). The centralized architecture of SDN is the bottle neck for scalable and dynamic nature of SDN.

II. SOFTWARE DEFINED NETWORK

SDN is the change in the underline of the network .this change in the architecture lead to make SDN programmable and flexible .SDN has two major characteristics separating the control plan from the data plan that and the centralized the control plan with the controller which manages the entire network and its components, such as OpenFlow switches, routers, and other, via Application Programming Interfaces (API) the SDN Controller uses a protocol called OpenFlow to control switches and routers from the controller

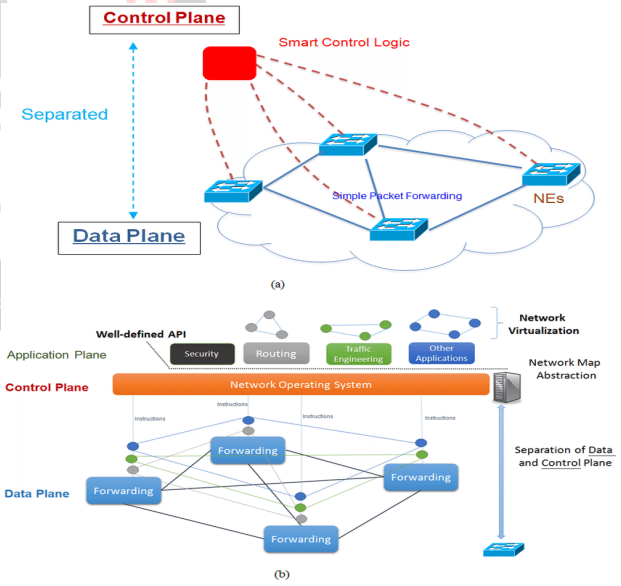

Fig. 1. SDN infrastructure

III. SDN CONTROLLER

The OpenFlow controller provides a programmatic interface to the OpenFlow switches. Using this programmatic interface, network applications, referred to as Net Apps, can be written to perform control and management tasks and offer new functionalities.

The control plane in SDN and OpenFlow in particular is logically centralized and Net Apps are written as if the network is a single system. While this simplifies the

 DOI : 10.35291/2454-9150.2020.0704

control, management, and policy enforcement tasks, the bindings must be closely maintained between the controller and OpenFlow switches.

The first important concern of this centralized control is the scalability of the system and the second one is the placement of controllers.

A recent study of the several OpenFlow controller implementations. Recalling the operating system analogy, an OpenFlow controller acts as a network operating system and should implement at least two interfaces: a southbound interface that allows OpenFlow switches to communicate with the controller, and a northbound interface that presents a programmable application programming interface (API) to network control and management applications. They are two type of the SDN controller Centralized controller Provides a centralized global view of the network which allows online reactivity to spurious changes in network states and simplifies the development of sophisticated functions and distribution controller A distributed controller provides fault tolerance, but requires an additional overhead to maintain the network state consistent across all the controllers. Hence, when the network state changes there will always be an inconsistency period of time.

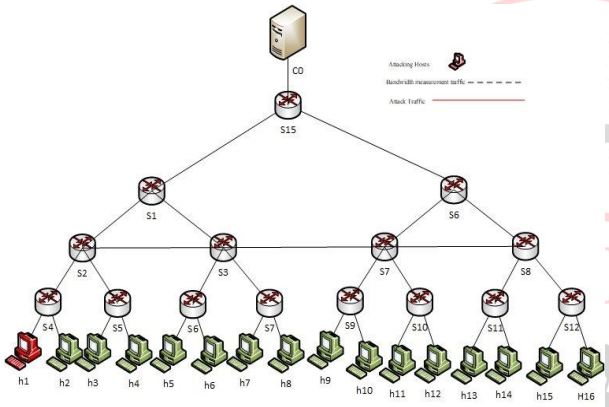

Fig. 2. SDN topology with centralized controller

IV. METHODOLOGY AND DESIGN

In this study we will implement the model using Mininet emulator to simulate the network environment and virtual BOX from oracle to handle the software's. In the virtual box we setup and run the controller (Floodlight) after study the behavior of the controller We will study the commands necessary to operate the controller in detail, in addition to that applying some commands that make the traffic dedicated and control the paths of the traffic through the network.

we try to find the suitable application for the controller to achieve proper performance of the network.

First step to design the topology we start with the virtual box as the coming follow.

4.1 Oracle VM Virtual Box

VM is a free, open source, cross-platform application for creating, managing and running virtual machines (VMs). Virtual machines are computers whose hardware components are emulated by the host computer.

It was acquired by Sun Microsystems in 2008, which was in turn acquired by Oracle in 2010.

It enables you to set up one or more virtual machines (VMs) on a single physical machine, and use them simultaneously, along with the actual machine. Each virtual machine can execute its own operating system, including versions of Microsoft Windows, Linux, BSD, and MS-DOS. You can install and run as many virtual machines as you like – the only practical limits are disk space and memory.

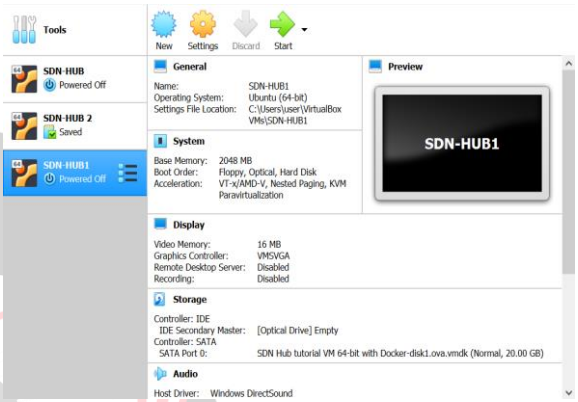

Fig.3. Oracle virtual Box interface

4.2 Mininet emulator

Mininet is an open-source network emulator which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking.

It creates a simulated network that runs real software on the components of the network so it can be used to interactively test networking software. It helps in creating complex custom scenarios using the Mininet Python API. Mininet is a system that provides large networks for rapid prototyping on a single computer. Using lightweight virtualization mechanism it creates scalable software-defined networks.

You can create a network with a single command. For example,

```
sudo mn --switch ovs --controller ref --topo tree,depth=2,fanout=8 --test pingall
```

starts a network with a tree topology of depth 2 and fanout 8 (i.e. 64 hosts connected to 9 switches), using Open vSwitch switches under the control of the OpenFlow/Stanford reference controller, and runs the pingall test to check connectivity between every pair of nodes. (This takes about 45 seconds on my laptop.)

 DOI : 10.35291/2454-9150.2020.0704

to interacting with a Network Mininet's CLI allows you to control, and manage your entire virtual network from a single console. For example, the CLI command

```
mininet> h2 ping h3
```

4.3 Setup and Running the Controller (Floodlight) from CLI

Floodlight is open source based on Java, and one of the most popular SDN controllers supporting physical and virtual OpenFlow switches.

The Floodlight architecture is including device management module, learning switch, counter module , load balancer, topology module, Web Graphical User Interface (Web GUI) for web access. Floodlight Provider module called as core module, handles I/O from network devices and turns incoming OpenFlow messages into Floodlight events. It uses LLDP protocol to discover network topology floodlight presents a RESTful API allowing some applications to learn and get the state of the controller and network [10].and floodlight controller uses event listeners for receive notifications.

Floodlight provides reactive and proactive applications: Java APIs for reactive and RESTful APIs for proactive application style, can use the RESTful APIs to get information about the network state. Floodlight RESTful API uses the Restlet framework and includes a small web server inside that allows external applications to communicate with the SDN controller.

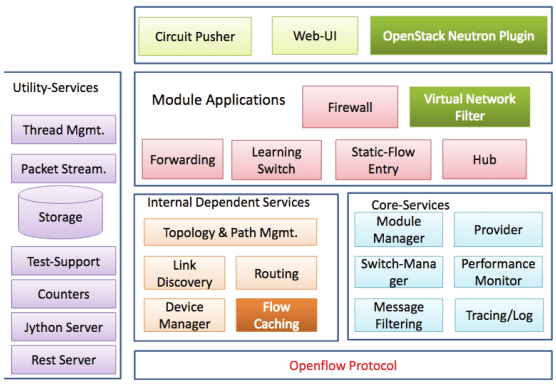

Fig. 4. floodlight instruction

You can manually choose which openjfx version which you tend to install and then hold it.

```
$ sudo apt install libopenjfx-java=8u161-b12-1ubuntu2 libopenjfx-jni=8u161-b12-1ubuntu2 openjfx=8u161-b12-1ubuntu2
$ sudo apt-mark hold libopenjfx-java libopenjfx-jni openjfx
$ cd floodlight
$ ant
```

After download the floodlight controller return to mininet and execute the following commands to run the floodlight

```
cd Desktop/
cd floodlight/
java –jar target/floodlight.jar
```

to connect the network with the floodlight controller

Sudo mn command is to create the network ,Controller=remote is to connect the network with the floodlight controller and determine the ip and the port number, topo=tree depth=3,fanout=2 command its tree topology consist of 7 switches and 8 hosts.

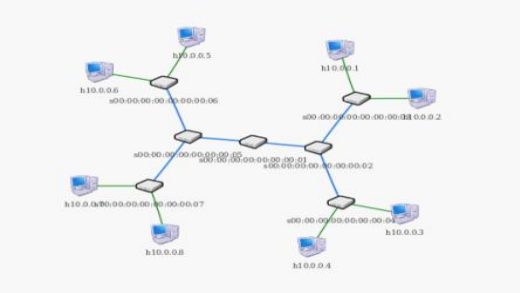

Fig. 5. tree topology

4.3 Running the Controller (Floodlight) from GUI

This GUI provides a way for users to view the controller's state information, to connect switches via inter-switch links and the hosts to the network, to monitor the flow tables of the switches and the network topology.

Most of the statistics can be queried and displayed in an easy-to-read and tabular fashion by using this web GUI. Additionally, several modules of the Floodlight Controller can be exposed to the end users via this web GUI.

For example, the Static Flow Pusher module has this GUI to insert the flows easily [4].

Fig. 6. in this network administrators can monitor the status, uptime, selected role of the controller, connected switches count, connected hosts count, and the links between the switches. the details about the controller's memory consumption, the storage tables, and all loaded modules can be monitored. All of these parts of the GUI uses the REST API calls of the Floodlight Controller.

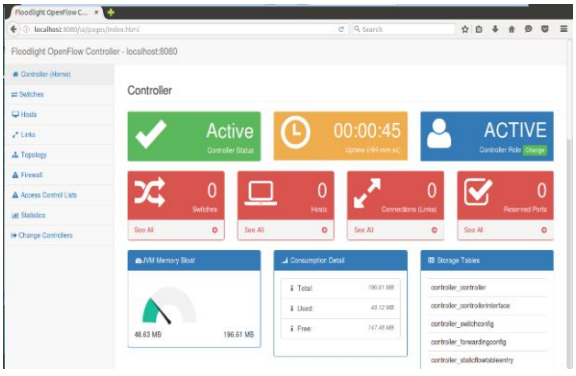

Figure .6. floodlight GUI home page

 DOI : 10.35291/2454-9150.2020.0704

V. EXPERIMENTS RESULTS (DEDICATE THE TRAFFIC)

In the practical side of this study, after running the controller and creating and designing the network , take the benefit from what SDN technology has made possible to manage and control any part of the network through the controller .

An efficient and fully functioning in network management is dedicating the path of the traffic from specific source to specific destination that can enable the network administrator to securing and optimizing the network infrastructure, and this was done using three different methods, which will be detailed in the following in the beginning stop the forward module by using the flowing command .

```
Sudo mn –controllernon –topo single
```

That mean the icmp will felt there is no reachability with hosts .

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X
h2 -> X X X
h3 -> X X X
h4 -> X X X
```

Fig.7. ICMP command

5.1 Dedicate the traffic base on switch port

A specific data path was specified based on the port number This method allowed not to send or receive from any other connected device on the same network, which guarantees to the network manager that the device connected to the specified port has no access except for the device specified for him in the aforementioned port.

It is also possible to allow data to be sent only without receiving it and by implementing the Only the first command to allow posting.

```
$ sudo ovs-ofctl -O Openflow13 add-flow s1 in_port=1,actions=output:2
$ sudo ovs-ofctl -O Openflow13 add-flow s1 in_port=2,actions=output:1
```

in the following commands we dedicate the traffic to pass through the port one and transmit data only to the port two only without communicate to any other hosts that connecting with him in the network . and the second command to transmit data on the opposite side from host (2) via port(2) to host (1)via port (1) .

5.2 Dedicate the traffic base on IP Address

In the following command, the data was controlled depending on the IP address, which allowed communication at the networks layer, and this is one of the advantages of the SDN and open flow protocol that it works on all layers.

using a simple topology that was shown in fig (5) host1 with (IP = 10.0.0.1) connect and send packet only to host 4 (IP = 10.0.0.4).

```
$ sudo ovs-ofctl -O Openflow13 add-flow s1 in arp,nw_src=10.0.0.3,actions=output:4
$ sudo ovs-ofctl -O Openflow13 add-flow s1 in arp,nw_src=10.0.0.4,actions=output:3
```

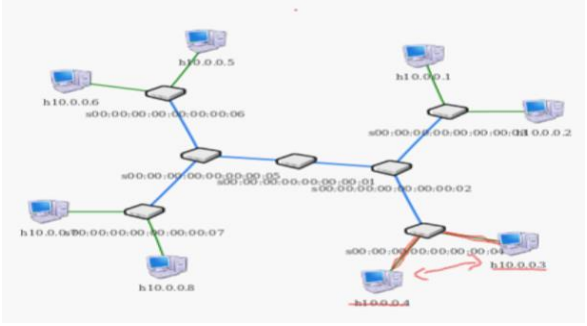

Fig.8. Dedicate the traffic base IP address

5.3 Flow entry base on MAC Address

In this part, the data path has been allocated depending on the mac, where the controller deals with the data link layer by recognizing the mac address of the source and the mac address of the destination.

Here the mac address of the first host (00:00:00:00:00:03) which will send and receive the traffic from and to the second host with mac address (00:00:00:00:00:04)

```
$ sudo ovs-ofctl -O Openflow13 add-flow s1 dl_src=00:00:00:00:00:03,actions=output:4

$ sudo ovs-ofctl -O Openflow13 add-flow s1 dl_dst=00:00:00:00:00:04,actions=output:3
```

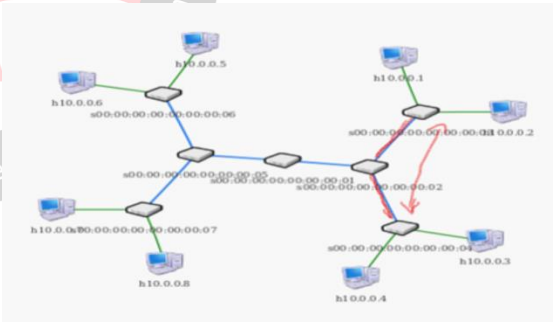

Fig.9. Dedicate the traffic base MAC address

after applied the all rules the hosts can connecting to gather and the ping command is reachable

Pingall command by using this command will making all network hosts transmit ping packets onto other in order to confirm the communication between them then the reachability. making utilizing the Internet Control Message Protocol.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

Fig. 10. ICMP command after applying flow entry rules

 DOI : 10.35291/2454-9150.2020.0704

VI. CONCLUSION AND FUTURE SCOPE

The use of SDN technology allowed maximum effectiveness in performance and the controller contributed to creating an ideal environment in determining the type of service and the beneficiary of it,

The Floodlight controller and GUI provide many useful tools and a programmable network framework.

As in this paper we have provided the clear idea how to create experimental test bed with steps of configuring and managing software define network via floodlight controller and a specific data path was also allocated depending on the port number, IP address, and mac address which gives the network privacy and more security to each host.

Further, we will come up with few more papers on implementation of other SDN controllers like ONOS and RYU in the coming future. we also has planned to compare the quality of services between different controllers of SDN, and we will try to apply high availability and load balancing by using multi controllers on SDN topology .

REFERENCES

[1] Benzekki, Kamal; El Fergougui, Abdeslam; Elbelrhiti Elalaoui, Abdelbaki (2016). "Software-defined networking (SDN): A survey". Security and Communication Networks. 9 (18): 5803–5833. doi:10.1002/sec.1737.

[2] Zhang, H., Cai, Z., Liu, Q., Xiao, Q., Li, Y., & Cheang, C. F. (2018). A survey on security-aware measurement in SDN. Security and Communication Networks, 2018.

[3] K. Ahokas, Software-defined networking, Aalto University CSE-E4430 Methods and Tools for Network Systems, Finland, Autumn 2014.

[4] R. Izard (administrator), H. Akcay (GUI developer), (2017). Floodlight WEB GUI. [online] Floodlight.atlassian.net. Available at: https://floodlight.atlassian.net/wiki/spaces/

[5] Pooja, M. Sood, SDN and Mininet: Some Basic Concepts, Int. J. Advanced Networking and Applications, 07(02), 2015, 2690-2693.

[6] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka and T. Turletti, A Survey of SoftwareDefined Networking: Past, Present, and Future of Programmable Networks, IEEE Communications Surveys & Tutorials, 16(3), 1617-1634.

[7] OpenFlow Switch Specification, Version 1.1.0 Implemented (Wire Protocol 0x02). [Online]. Available: http://www.openflow. org/documents/openflow- spec-v1.1.0.pdf

[8] Shi, Lei, Bin Liu, Changhua Sun, Zhengyu Yin, Laxmi Bhuyan, and H. Jonathan Chao. "Load-balancing multipath switching system with flow slice." Computers, IEEE Transactions on 61, no. 3 (2012): 350-365.

[9] Prete, Luca, Fabio Farina, Mauro Campanella, and Andrea Biancini. "Energy efficient minimum spanning tree in OpenFlow networks." In Software Defined Networking

(EWSDN), 2012 European Workshop on, pp. 36-41. IEEE, 2012.

[10] Dixit, Advait, Fang Hao, Sarit Mukherjee, T. V. Lakshman, and Ramana Kompella. "Towards an elastic distributed SDN controller." In ACM SIGCOMM Computer Communication Review, vol. 43, no. 4, pp. 7-12. ACM, 2013.

[11] Kim, Hyojoon, J. R. Santos, Y. Turner, M. Schlansker, J. Tourrilhes, and Nick Feamster. "Coronet: Fault tolerance forsoftware defined networks." In Network Protocols (ICNP), 2012 20th IEEE International Conference on, pp. 1-2. IEEE, 2012.

 DOI : 10.35291/2454-9150.2020.0704