International Journal of Advanced Technology in Engineering and Science
Vol. No.5, Issue No. 04, April 2017
www.ijates.com
ijates
ISSN 2348 - 7550

EMULATING SOFTWARE DEFINED NETWORK USING MININET

Manamrit Singh Sroya[1], Vikramjit Singh[2]

[1,2] *Computer Science and Engg, NorthWest Group of Institutions,Dhudike, Moga (INDIA)*

ABSTRACT

Software defined Network is a new approach in the area of computer network that totally changes the model of network. In this new model of network control plane and forwarding plane are to be separated. Research on this topic is on going now days but there are not many devices are available that contain SDN functionality. If devices are available, these devices are very costly. Mininet is a tool by which researchers can do experiments with negligible cost and great features. The main advantages of mininet is that code that we run in mininet we can easily run that code in real environment. Main focus of this paper is on the study of mininet tool and evaluation of the tool. Mininet is an open source emulation software tool that is used for learning and testing Software Defined Network. To test our Software Defined Applications we need a virtual network consist of Hosts, switches, controllers. We can launch this type of network with single command in mininet. Mininet installed on a single computer or any virtual machine that has constrained with limited resources. Instead of Mininet we can also use hardware testbed that are accurate, fast but more financial cost. One another option is to use simulators that have low cost but slow in speed and also require modification in the code. So mininet is best option due to ease to use, scalability, better performance than simulators and real testbeds.

Keywords*: *Mininet, OpenFlow, Software Defined Networking, Ryu Controller, Python.

I. INTRODUCTION

Mininet Internet become a communication infrastructure due to large number of users and each having different applications that are also increase continuously. But the internet is still unable to handle the challenges such as network control and configuration issues and flexibility with its capabilities. These are interesting topics for research in these days. Project Works are also initiated in these topics.

In traditional network replacement of network devices or modification in the behavior of existing devices are very difficult task due to tight bound between control and forwarding parts. The researchers need such solution that enables the users to use a network with less need of replacement of network devices [1]. So in the change of technology we can smoothly add elements into the network with negligible cost.

Software-Defined Networking (SDN) is a new architecture or model in which control plane and data plane are decoupled which are tightly bound in traditional devices as shown in Fig. 1. Such a network is flexible and dynamic that is big advantages in internet. Openflow is standard protocol for communication between data plane and control plane. The decision of route of packet is taken by the openflow protocol with the help of network applications [2].

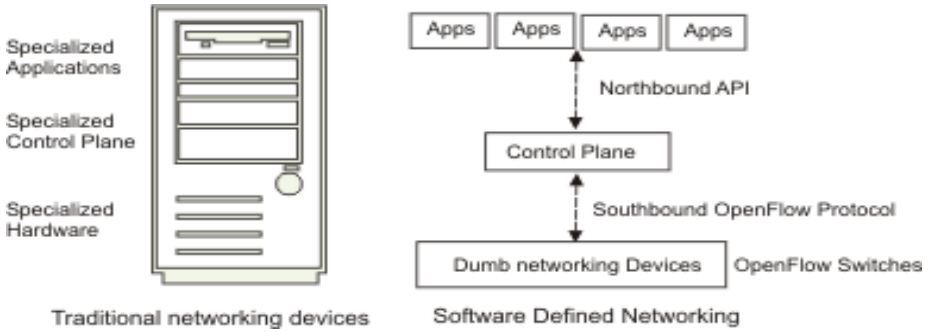

Fig. 1 traditional vs software defined network

The second section of this paper describe about Software Defined network and its motivation, different components of SDN and how these components are interact with each other in the network. The third section discusses about emulation tool called Mininet, how we can create a network using mininet, basic commands that are run on mininet and also elaborate various mininet topologies and then write about how we can add rules in flow table of switch using ofctl. The fourth section described about how we can run different SDN applications on mininet with RYU controller. The last section presents the conclusion of the paper and some suggestions for future works

II. SOFTWARE DEFINED NETWORKING

In recent years, Software Defined Network (SDN) is a technology that is mostly used to manage the computer network. In this architecture, both planes (control & data) are separated in network devices that are closely coupled in traditional network devices as shown in Fig. 2. The functionality of control plane performed by controller and functionality of data plane performed by switch. The task of switch is only forwarding the packet. Due to separation of planes, it increases the scalability of network, reducing cost of network elements, easily implementation of new ideas [3].

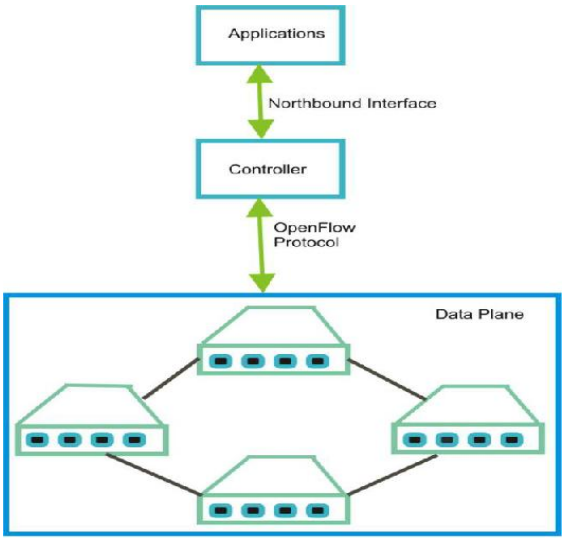

Fig. 2 SDN architecture

Communication between planes is possible with the help of Openflow protocol. Network administrator has a central control of the whole network. Now the forwarding devices are dumb devices that contain no

functionality. controller inserts the flow rules into flow table of switch with the help of this protocol. Openflow protocol has a programming interface to add or remove flow entries into flow table of switch [4]. When a packet comes, switch has no information about that packet, the decision is taken by the controller according to applications.

III. MININET TOOL FOR TESTING SDN NETWORK

There are various types of Virtualization are available in these days such as platform virtualization, network virtualization, hardware virtualization. Mininet is a network virtualization that is used to create a virtual SDN network consist of components such as number of hosts, multiple openflow enabled switches, different controllers and links between these components on a single computer as shown in Fig. 3.

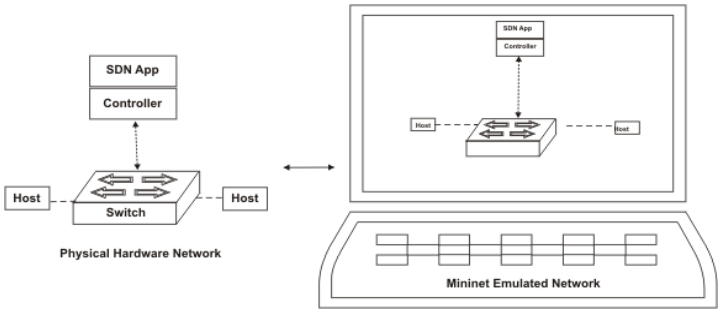

Fig. 3 mininet emulator

Firstly, Mininet was developed by number of professors at Stanford University for research purpose. Now these days, this is widely used by SDN researchers due to its great features. We can use default topologies to create a SDN network. We can also create custom topologies with the help of python code. Different types of commands are used to create a switches, hosts and controller. Such as type "mn" command to create 1 openflow enabled switch and 2 hosts that are attached with this switch. The controller in this case is reference controller is also added [5].

3.1 Create SDN network components using mininet

SDN enable researchers to create network elements such as hosts, switches, controllers, links and also customize these elements. The sharing of network elements with other network is also possible means that you can share a virtual mininet machine of your computer with other researchers. A host that is created in mininet is a real linux system that is basically a light weight virtualization [6]. The OpenFlow enabled virtual switches created by Mininet is same as openflow enabled hardware switches. When we run topology with none controller then by default reference controller is running. We can also pass remote controllers such as POX [7], RYU [8], Floodlight controller [9].

3.2 Mininet basic commands

There are different types of commands are available to interact with mininet. Some important commands are given below:

3.2.1 nodes

It is used to checking list of nodes available such as name of hosts, switches, and controllers. (Fig 4).

3.2.2 net

It shows all network links means connection between each components. (Fig 4).

3.2.3 dump

It shows information about the nodes, switches and controllers and process id of each node. (Fig 4).

3.2.4 help

It gives a list of available commands

3.2.5 ifconfig

Run this command to check the IP address of a particular host. (Fig 4)

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet>
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet>
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1527>
<Host h2: h2-eth0:10.0.0.2 pid=1530>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=1535>
<Controller c0: 127.0.0.1:6633 pid=1520>
mininet>
mininet> h1 ifconfig
h1-eth0   Link encap:Ethernet  HWaddr 76:67:31:2a:39:04
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Fig. 4 running of some basic mininet commands

3.2.6 ping

It shows the connectivity between hosts such as h1 ping h2 show connection between these two hosts.

3.2.7 pingall

It shows the connectivity between all hosts in the network.

3.2.8 iperf

It is used to check bandwidth between hosts.

3.2.9 exit

Use this command to end the topology.

3.2.10 xterm

Used to open different window for each node.

3.2.11 time

Measure time taken for commands in Mininet.

3.3 Mininet Topologies

There are 5 default topologies such as minimal, single, reversed, linear, tree that are mostly used. But according to our needs we can also create custom topologies. Different topologies are discussed here.

3.3.1 Minimal Topology

If we run command **"mn"** it create I openflow switch and 2 hosts connected with the openflow switch as shown in Fig. 5

3.3.2 Single Topology

if we run command **"mn --topo single, 6"** it create 1 openflow switch and 6 hosts connected with the openflow switch as shown in Fig. 5

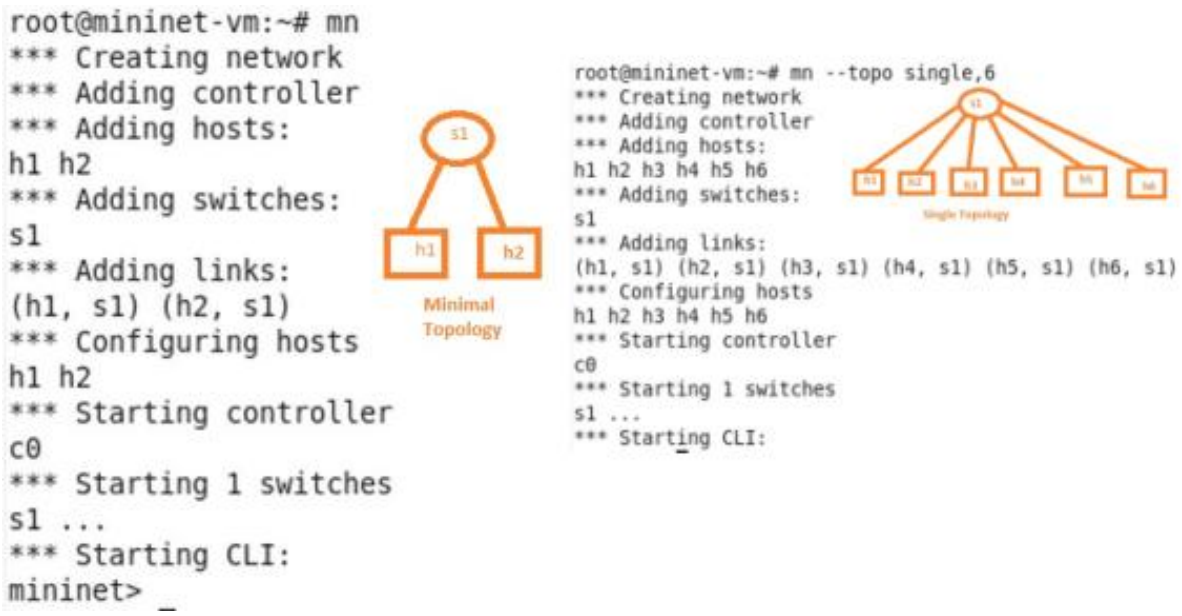

Fig. 5 minimal and single topology

3.3.3 Reversed Topology

It is similar to single topology but having connection in reverse order.

3.3.4 Linear Topology

if we run command "mn --topo linear, 2" it create 2 openflow switches and 1 host connected with the each openflow switch as shown in Fig. 6.

3.3.5 Tree Topology

If we run command "mn --topo tree, 2" it create switches at two levels. At level 1, s1 switch is added and at level 2, s2 and s3 switches are added. Two hosts are attached to every lower-level switch such as s2 and s3 as shown in Fig. 6.

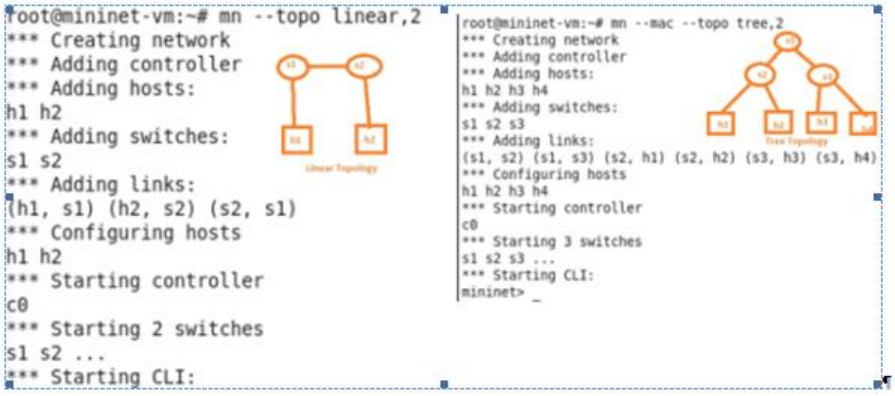

Fig. 6 linear and tree topolo

3.4 Add flow rules into flow table of switch using ofctl

The ofctl is a utility that is used to add flow rules into flow tables of openflow switch . We can control the

functionality of flow table using ofctl. If we want that our openflow switch Act as switch then by using ofctl utility we can add rules into flow table of switch.

ofctl add-flow s1 in_port=1,dl_dst=00:00:00:00:00:03, actions=output:3

According to rule all the packets coming from port 1 and destination mac address 00:00:00:00:00:03 will be sent to port 3. If we enter a rules using ofctl we have no need of any controller. When a packet come from port 1 and reach at openflow switch, according the flow table rule the switch forwarded the packet to port 3.

VI. RUN SDN APPLICATIONS ON MININET WITH RYU CONTROLLER

With the help of ofctl we can manually add the flow rules into openflow switch. Disadvantage of ofctl is that we have to configure each switch one by one. On the other hand controller visualizes and controls the whole network from central point. Openflow provide a connection between switches and controllers such as Ryu, POX, Floodlight. Now openflow switch consult to the openflow controller to add a flow rules into the flow table of switch when there is no flow rule and packet are come from host. Then openflow controller according to the application, insert rules into the flow table of switch then openflow switch perform that particular action. In our case we are using Ryu controller for communication with the openflow switch. The applications that we run in Ryu are written in python language. Ryu provides its users a well defined API to write different types of applications [10].

In our first test we create a linear topology with four hosts that are connected with single switch

To create a such topology type this command

mn --mac --topo single,4 --controller remote

On the another terminal enter into ryu directory and then execute the following command to run the Ryu controller

./bin/ryu-manager ryu/app/simple_switch.py

For testing send icmp packets from host h1 to h3. For this type the following command.

mininet> h1 ping h3

The test was successful there is no loss of packets. When packet come from host h1, openflow switch contain no flow rule about how to handle this packet. Openflow switch send the packet to ryu controller. Then there are two approaches how ryu controller informs the openflow switch about the packet. First approach is Ryu controller directly send the packet to end host. But in this case our every packet goes the controller and it has performance issues such as congestion on the controller. Second approach is that ryu controller inserts the rules into flow tables then same packets are handled by openflow switch locally. We are using second approach due to better performance. If we disconnect the controller then all the packets were lost because when packet come into switch, switch tried to send packets to controller but there is no controller then all packets were lost.

V. CONCLUSION

Software Defined Network is a new paradigm in the field of computer network. It has gaining a great attention from the researchers due to separation of hardware from the underlying software. Mininet is tool used for the purpose of research in the field of SDN. Alternatives of mininet are also possible such as simulators and real hardware. Speed of real hardware is better than mininet but very expensive. We can also use simulators that are

less expensive than real hardware but code that we run on it, same code is not run on the real hardware. So we conclude that Mininet is better from others for research purpose. But there are some disadvantages that we can't create very large network with mininet because it is limited with resources of single computer or laptop.

REFERENCES

[1] Nunes, Bruno Astuto A., Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. "A survey of software-defined networking: Past, present, and future of programmable networks." *IEEE Communications Surveys & Tutorials* 16, no. 3 (2014): 1617-1634.

[2] Hu, Fei, Qi Hao, and Ke Bao. "A survey on software-defined network and openflow: From concept to implementation." *IEEE Communications Surveys & Tutorials* 16, no. 4 (2014): 2181-2206.

[3] Xia, Wenfeng, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, and Haiyong Xie. "A survey on software-defined networking." *IEEE Communications Surveys & Tutorials* 17, no. 1 (2015): 27-51.

[4] Feamster, Nick, Jennifer Rexford, and Ellen Zegura. "The road to SDN: an intellectual history of programmable networks." *ACM SIGCOMM Computer Communication Review* 44, no. 2 (2014): 87-98.

[5] Lantz, Bob, Brandon Heller, and Nick McKeown. "A network in a laptop: rapid prototyping for software-defined networks." In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, p. 19. ACM, 2010.

[6] Kaur, Karamjeet, Japinder Singh, and Navtej Singh Ghumman. "Mininet as Software Defined Networking Testing Platform." In *International Conference on Communication, Computing & Systems (ICCCS*. 2014.

[7] Kaur, Sukhveer, Japinder Singh, and Navtej Singh Ghumman. "Network programmability using POX controller." In *ICCCS International Conference on Communication, Computing & Systems, IEEE*, no. s 134, p. 138. 2014.

[8] Khondoker, Rahamatullah, Adel Zaalouk, Ronald Marx, and Kpatcha Bayarou. "Feature-based comparison and selection of Software Defined Networking (SDN) controllers." In *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*, pp. 1-7. IEEE, 2014.

[9] Wallner, Ryan, and Robert Cannistra. "An SDN approach: quality of service using big switch's floodlight open-source controller." *Proceedings of the Asia-Pacific Advanced Network* 35 (2013): 14-19.

[10] Shalimov, Alexander, Dmitry Zuikov, Daria Zimarina, Vasily Pashkov, and Ruslan Smeliansky. "Advanced study of SDN/OpenFlow controllers." In *Proceedings of the 9th central & eastern european software engineering conference in russia*, p. 1. ACM, 2013.