Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 07, July-2019

# Comparative Analysis of Performance of Controllers in Software Defined Networks using Mininet

Maria George
Department of Computer Science
CHRIST(Deemed to be University),Hosur Road
Bangalore-29,Karnataka

Deepa V Jose
Department of Computer Science
CHRIST(Deemed to be University),Hosur Road
Bangalore-29,Karnataka

***Abstract*—This Software Defined Networks, the latest trend in networking architecture aims to provide agile and flexible networks. It is one of the most highlighted research areas in networking. As the number of nodes increases in the network it is difficult or sometime impossible to handle all the nodes which are connected to same or different network in different scenarios. This was one of the limitations in the traditional networking. This research work aims to analyze and test the capability of the controllers to come up with a solution to this issue. The performance analysis is done based on scalability and throughput on different number of nodes and varying topology scenarios. From among the multiple existing controllers that provide Software Defined Networks functionalities, two of the best choices- the Beacon Controller and the Floodlight Controller are used along with Mininet for performance analysis using simulation.**

***Keywords—Beacon;Controller;FloodLight;Mininet; Networking; OpenFlow;Software Defined Networks; Simulation***

## I. INTRODUCTION

Computer Networks is a complex collection of heterogeneous network devices interconnected with each other which enable data communication from anyone to anywhere at any time. With the internet services, more devices such as switches, routers, firewalls etc. also gets into this network. In this scenario, it is often hectic for the network operators to configure the network with various high-level policies and respond to wide range of network requirements that may occur.

Technology has made a lot of innovation the field of networks. One among them is the Software Defined Networking (SDN). The term SDN was originally defined to represent the ideas and work around OpenFlow, the communications protocol that gives access to the forwarding plane of a network switch or router over the network.

Since the number of users and in turn the number of devices has drastically increased along with the time, some major problems emerged to occur like configuring each system, decentralization, difficulty in reprogramming devices etc. These issues are very critical and is a time consuming process. This was one of the drawbacks of traditional switching in which the reprogrammability of switches is not applicable.

The control plane and data plane is coupled together in switches which leads to the introduction of SDN. So due to this reason SDN helps in centralizing the devices and programmability became easy which reduced the cost and increased time efficiency. Some of the popular SDN controllers are POX[1],Ryu[2],OpenDayLight[3],NOX[4],ONOS and so forth. They would manage and configure the available switches dynamically according to the necessity of the user. These controllers would control all the operations for the forwarding of the packets from the source to the destination using interfaces like NorthBound API and SouthBound API. NorthBound API such as REST, acts as a interface between the Application Layer and Control Layer which is used for the implementation of business policy over application layer. This also uses service policy to state traffic behaviour. The interface between the Control Layer and the Infrastructure Layer is the Southbound API that has got a forwarding rule after installation of the controller. Some of the southbound API are Opflex [5], OpenFlow [6], NETCONF[7], POF [8], ForCES [9] etc.

SDN has got an architecture which comprises of mainly three layers. They are Application Layer,Control Layer and Infrastructure Layer as mentioned in Fig 1. The Application Layer can program explicitly in this layer to communicate with the network. Further it also helps to get the abstract view of the network by collecting the data from the control plane for taking decisions. It consists of an abstract view of business applications to program explicitly. Control Layer is the logical entity where all the instructions are received from other networking components. It has the controller who can control and extract information about the network from the hardware components and make communication possible. Infrastructure layer consists of switches that is used to forward the packets and has an inbuilt flow table to check the incoming and outgoing packet from one host to another. A southbound API connects the controller and the switch and the Northbound API connects the controller and the applications to make communication possible.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 07, July-2019

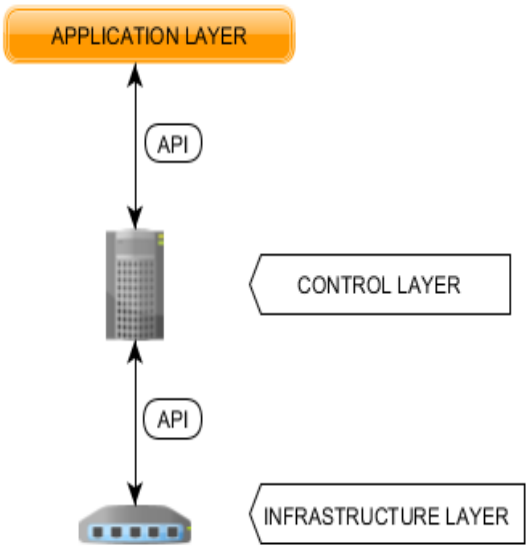



Fig 1: SDN Architecture

SDN is one of the widely used technology that has been used for home[10] and by companies such as Google, CISCO, Facebook etc. that decouples the control plane from the data plane and are kept for different scenario on scalability by a remote system. To analyze this scalability issue and its performance certain experiments are taken to evaluate by comparing the two controllers. This paper is organized as follows. An outline of the related work is mentioned in Section 2.Section 3 gives an overview of Floodlight controller and beacon controller. Section 4 deals with the simulation test on scalability with certain scenarios to analyze which one among the controllers is more scalable. Section 5 elaborates the obtained experimental results and gives evaluation of the performance followed by conclusion and future scope.

## II. RELATED WORK

Erel et al.[11] did a Mininet-based simulation that improved the total flow throughput and scalability of the overall network.The authors in their demonstration have used OpenDaylight controller to simulate flow admission control module , OpenFlow version 1.3. for communication between separated Data and Control plane and Linux based operating system to build Mininet 2.1.0 are deployed in the simulator environment.

Sidki et al.[12] proposed an approach for the issue of fault tolerance by using a slave controller architecture with local mechanisms of virtual controller redundancy and synchronization between the controllers. The authors claims that this proposed approach enabled the network to cope with control plane crashes in the controllers without changing the OF protocol between controllers and switches.Tatang et al.[13] presented SDN-GUARD, a novel system for detecting and mitigating SDN rootkits. The basic idea is to perform a dual-view comparison that detects malicious network programming attempts. According to the authors ,the proposed approach is more effective and flexible in terms of application, and has less performance overhead.

Khorsandroo and Tosun[14]introduced a testbed and investigated SDN controller live migration in a virtual data center. It identifies container size, traffic volume, traffic pattern and transport layer protocol throughput as contributing factors of a successful SDN controller live migration. It then clarifies how these factors may affect a live migration process in terms of migration time and downtime through conducting experiments on a state-of-the art cloud data center testbed.

## III. OVERVIEW OF FLOODLIGHT CONTROLLER AND BEACON CONTROLLER

Floodlight[10] controller is a part of floodlight project that helps the beginners to expertise in the field of SDN. It is a Apache licensed, Java based OpenFlow controller and one of the momentous contribution from Big Switch Network, developed by an open community of developers and has got a user-friendly GUI.This helps to easily understand and create the connectivity link, number of switches, number of hosts and to make the controller active or inactive. The architecture of Floodlight is given in Figure 2.It can handle OpenFlow and non OpenFlow networks and multiple hardware switches. An http REST command is used to interact with the controller and to retrieve information and services.

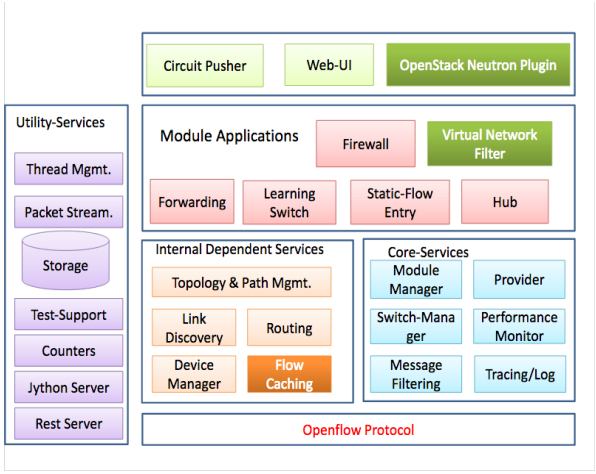


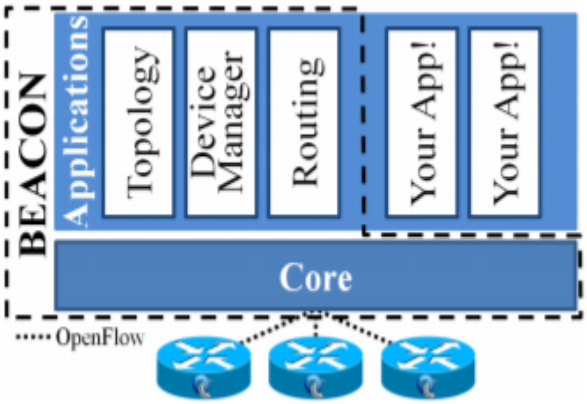

Fig:2 Architecture of FloodLight

The architecture of Floodlight consists of internal and utility services which contains various modules. One such module is Topology and path management where the computation of the shortest path is done using Dijkstra's algorithm The Link discovery module is responsible for maintaining the link state information by using LLDP packet. Routing module routes the packets from source to destination. Device manager keep track of all the source storage and network nodes. Forwarding is a module that forwards the packets of applications and many more.

Beacon is a fast, cross-platform, modular, Java-based OpenFlow controller that supports both event-based and threaded operation. Beacon has been in development since early 2010, and has been used in several research projects, networking classes, and trial deployments. It currently powers a 100-vswitch, 20-physical switch experimental data center and has run for months without downtime. It's also a Java based cross platform and runs on many platforms from high end multi-core Linux servers to Android phones. Beacon's goals are to improve developer productivity, provide the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 07, July-2019

runtime ability to start and stop existing and new applications, and to provide high performance. Developer's productivity is a definition of design and architectural choices with the aim of enabling developers to expand their time spent productively developing applications. Runtime modularity is an implementation supporting starting and stopping both existing and new applications from a running beacon instance. Performance is designs considered for the read and write paths of Beacon, resulting in a multithreaded implementation with linear performance scaling. The beacon architecture is given in Fig 3.




In beacon architecture, the device manager tracks all the devices that are seen in the network topology which consists of IP address, ethernet, switch and port last seen etc. It's an interface to search for new devices that will register with the network and updates the information. Topology is used to find the connection between the OpenFlow switches and also retrieves the information of every links being connected. Routing is a module that provides the shortest path between two devices in the network. It depends on the topology and device manager.

## IV. SIMULATION ENVIRONMENT

For doing the performance evaluation of Floodlight and Beacon controllers different scenarios are created. A custom topology has been created with ten different scenarios by incrementing the number of nodes. Mininet [11] is the simulator being used and as controllers Floodlight and Beacon. Mininet is installed in Ubuntu as in dual boot system which connects remotely to the Floodlight controller and Beacon controller one at a time. A python script is written for the customized topology with specific number of switches and increment in the number of hosts. These hosts are connected to the switch and the switches are connected to these controllers. The default switch of the Mininet has been used with the OpenFlow protocol. To evaluate the performance statistics a custom linear topology is implemented over 5 switches with ten different scenarios where these 5 switches are connected to each other.

Scenario A.20 hosts are connected to each switch(Total of 100 hosts + 5 switches+1 controller).
Scenario B.40 hosts are connected to each switch(Total of 200 hosts + 5 switches+1 controller).
Scenario C.60 hosts are connected to each switch(Total of 300 hosts + 5 switches+1 controller).
Scenario D.80 hosts are connected to each switch(Total of 400 hosts + 5 switches+1 controller).
Scenario E.100 hosts are connected to each switch(Total of 500 hosts + 5 switches+1 controller).
Scenario F.120 hosts are connected to each switch(Total of 600 hosts + 5 switches+1 controller).
Scenario G.140 hosts are connected to each switch(Total of 700 hosts + 5 switches+1 controller).
Scenario H.160 hosts are connected to each switch(Total of 800 hosts + 5 switches+1 controller).
Scenario I.180 hosts are connected to each switch(Total of 900 hosts + 5 switches+1 controller)
Scenario J.200 hosts are connected to each switch(Total of 1000 hosts + 5 switches+1 controller).

This same scenario from A to J is taken for the Beacon controller with 5 switches and 1 controller. The performance analysis has been taken for all these scenarios. Mininet has got inbuilt features of NOX controller that supports all the basic functionalities. In this paper, the comparative performance analysis of both Beacon and Floodlight controller is implemented. The default controller is not used for this experiment. The version of the floodlight used is 1.2 and for the beacon is 1.0.4. Fig 4 and 5 shows the creation of nodes. It is done by connecting a linear custom topology which calculates the minimum and maximum throughput while transmission of the data packets using TCP transmission. In Fig.5 the different scenarios is taken with 6 switches in a linear custom topology where all the nodes are connected to the switches. There is no direct connection between each node.

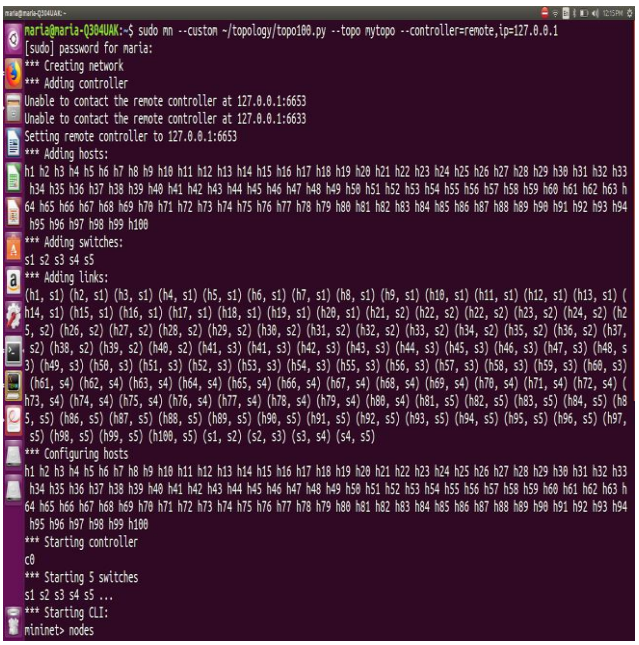
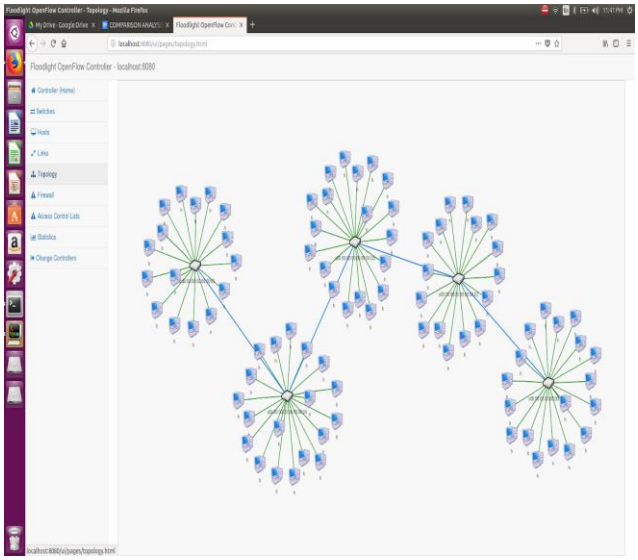



Fig 4: Creation of nodes

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 07, July-2019



The purpose for generating this scenario is to analyze the scalability issue and to choose which controller is more scalable depending on the situation and the performance traffic is been evaluated using TCP flow. This flow generation is simulated using Iperf for analyzing the throughput. Iperf is the tool which is used to measure the parameters like transfer rate, bandwidth, duration, packet delivery ratio, packet drops etc. For this research work the parameter used for comparing the controllers are throughput and scalability. The communication of packets in each scenario is between the first node and the last node of the network topology. Iperf is the tool used to get the desired result and this result is being plotted using the tool GNUplot. Filtering of the data is done using the 'grep' and 'awk' commands for the required parameters.

## V. PERFORMANCE ANALYSIS

The comparison of controllers was done for those different test scenarios to analyze the throughput with respect to the traffic network. The parameter that is chosen for comparison is throughput. To obtain the accurate throughput, TCP flow is used and is compared with the performance analysis of both controllers. Fig 6 and 7 displays the resultant graph for the given scenario. The switches are connected to each host and no hosts are directly connected to each other. The plotting is done using the tool GNUplot. According to the statistics of floodlight controller the average throughput is in the range 2.6 GB to 6.3 GB. According to the graphs the throughput is stable in the case of Floodlight controller which consists of hundred nodes in the linear custom topology. The communication is happening between these hosts virtually from a client to server. In the Fig 8 and 9 same parameters are taken with 300 number of nodes connected with five switches. According to the graph throughput is between 3.5 to 5.2 GB. The simulation executes in 150 seconds. In majority of the scenarios the throughput is stable even though the number of hosts is increased. By considering the third graph the scenario is with 500 numbers of hosts with the fixed number of switches. It is in the range of 2.6 to 5.2 GBs for the same duration of time. The stability is not much varied compared to the above graphs. Generally throughput will decrease when the number of nodes connected with all the switches in the network. It happens because there will be a heavy traffic when the packets flows for a long time.

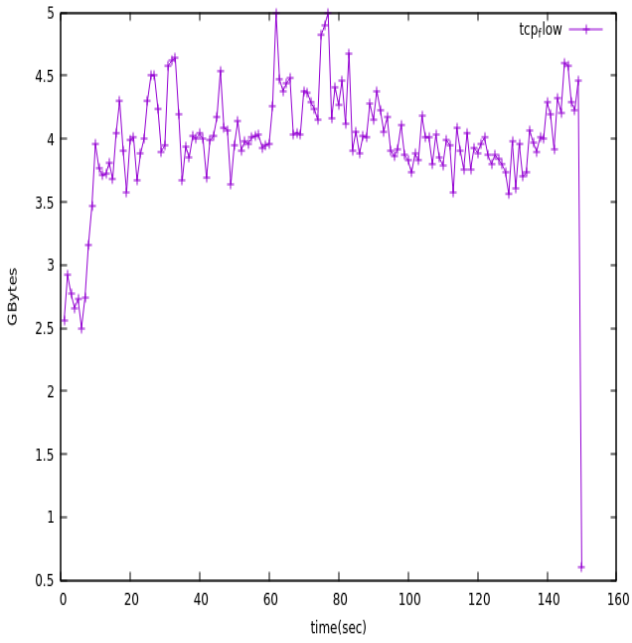



Fig 6: Throughput of Floodlight Controller( Scenario 1)

In Fig 8 and 9 the performance analysis of Beacon controller that was experimented with the same scenarios of the Floodlight controller but with another controller. According to the graphs it's clearly visible the stability of the controller with respect to the topology is less. In the first graph the throughput ranges from 3.4Gbytes to 5.3 Gbytes.The communication takes place with the TCP from the first node to the last node that are connected to the network with these 5 switches.

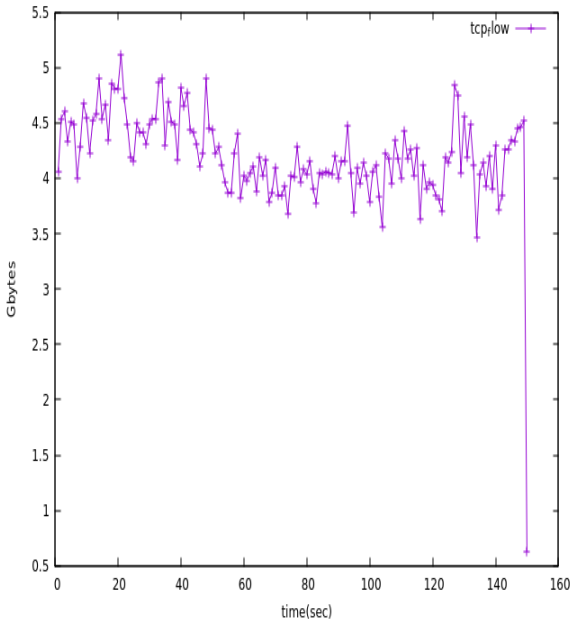



Fig 7: Throughput of Floodlight Controller( Scenario 2)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 07, July-2019

It was observed that second graph of fig.6 with 300 number of host and 5 switches the packets dropped at the 50th second at the rate of 3.5GB and at the same time the rate increased to 4.9 GB. There is a huge change in the transfer rate of data which has less stability. Even in the first scenario the first packet drop happens in the 40th second and also increased at the same time. The variation is much higher in the case of Beacon controller.

In the last graph of fig.6 the packet drops at the 15 second where the time interval is decreasing with respect to the increment of the hosts at the rate 3.3 GB. In only the 600 and 900 number of nodes the stability increases rest of the scenarios the stability is very less according to this topology. This happens due to the optimal usage of bandwidth when the communication is connectionless and also within the same network

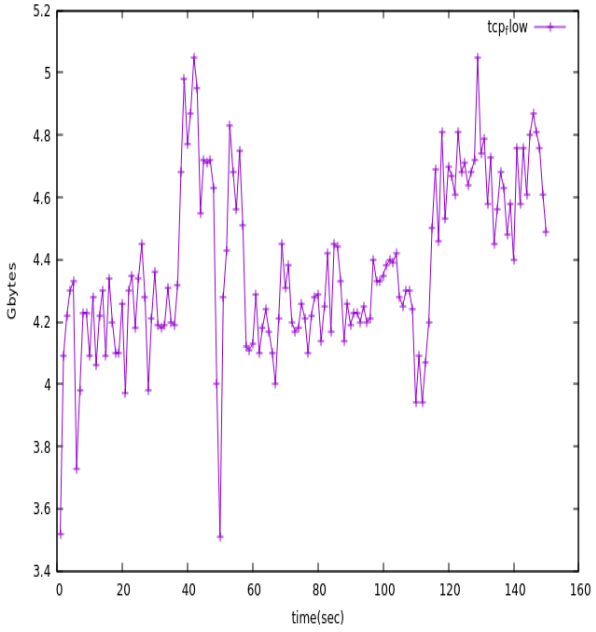



Fig 8: Throughput of Beacon Controller( Scenario 1)

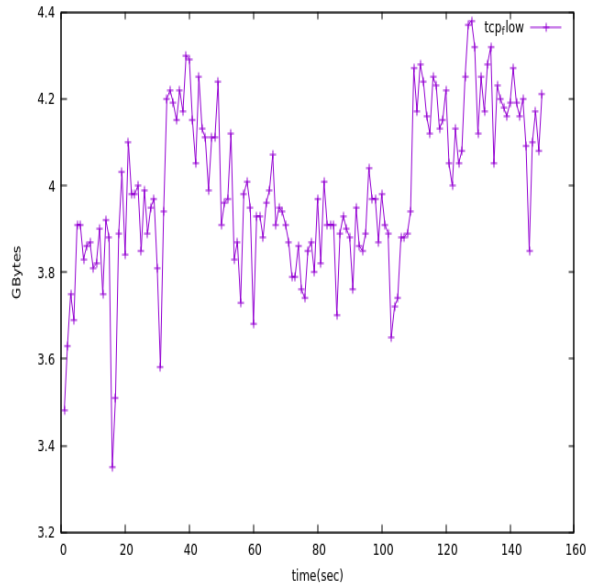



Fig 9:Throughput of Beacon Controller( Scenario 2)

## CONCLUSION AND FUTURE SCOPE

According to the simulation results, Floodlight controller is more scalable than the Beacon controller for the various scenarios implemented. In the simulation environment the scalability features of Floodlight and Beacon controller is clearly visible. A comparative analysis of the various other controllers available and optimal placement of the controllers will be the future work.

## REFERENCES

[1] McCauley, M. (2012). POX, http://www.noxrepo.org/
[2] Nippon Telegraph and Telephone Corporation, RYU network operating system, 2012, http://osrg.github.com/ryu
[3] OpenDaylight, Linux Foundation Collaborative Project, 2013,http://www.opendaylight.org
[4] Gude al, N. (2008). NOX: Towards an operating system networks. ACM SIGCOMM Computer Communication Review,vol. 38, no. 3, pp. 105–110.
[5] Smith M. (2014). OpFlex control protocol, Internet Engineering Task Force, http://tools.ietf.org/html/draft-smith-opflex-00
[6] McKeown , "OpenFlow: Enabling innovation in campus networks," ACM SIGCOMM - Computer Communication Review, vol. 38, no. 2, p. 69–74, 2008.
[7] Enns, R., Bjorklund, M., Schoenwaelder, J., Bierman, A. (2011). Network configuration protocol (NETCONF). Internet Engineering Task Forc,http://www.ietf.org/rfc/rfc6241.txt
[8] Song, H. ,Protocol-oblivious forwarding: Unleash the power of SDN through a future- proof forwarding plane.Proceedings of ACM SIGCOMM Workshop Hot Topics Software Defined Netw II. p. 127–132, 2013.
[9] Doria et al, Forwarding and control element separation (ForCES) protocol specification. Internet Engineering Task Force. (2010), http://www.ietf.org/r/fc/rfc5810.txt.
[10] Project Floodlight, Floodlight. (2012),http://floodlight.openflowhub.org/
[11] Mu¨ge Erel, Emre Teoman, Yusuf O¨ zc¸evik, Go¨khan Sec¸inti, and Berkk,Scalability Analysis and Flow Admission Control in Mininet-based SDN Environment, IEEE Conference on Network Function Virtualization and Software Defined Networks , 2015.
[12] Liran Sidki , Yehuda Ben-Shimol , Akiva Sadovski." Fault Tolerant Mechanisms for SDN Controllers", IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2016.
[13] Muge Erel, Emre Teoman, Yusuf Ozc¸evik, Gokhan Sec¸inti, and Berk Canberk," Scalability Analysis and Flow Admission Control in Mininet-based SDN Environment," EEE NFV-SDN - Third International Workshop on Security in NFV-SDN,2017.
[14] Sajad Khorsandroo,Ali Saman Tosun ," An Experimental Investigation of SDN Controller Live Migration in Virtual Data Centers, "IEEE NFV-SDN 2017 - Workshop on Federated Testbeds for NFV/SDN/5G: Experiences and Feedbacks, 20