Performance Evaluation of OpenDaylight SDN Controller

Zuhran Khan Khattak, Muhammad Awais and Adnan Iqbal
Department of Computer Science
Namal College
Mianwali, Pakistan
Email: {zuhran2010,awais2010,adnan.iqbal}@namal.edu.pk

***Abstract*—Enourmous amount of data has resulted into large data centres. Virtual machines and virtual networks are an integral part of large data centres. As a result, software defined network controllers have emerged as viable solution to manage such networks. The performance analysis of network controllers is generally carried out through benchmarking. Although several benchmarking studies exist, recently launched OpenDaylight SDN Controller is not considered in any benchmarking study yet. In this paper, we present initial results of benchmarking of OpenDaylight SDN Controller with well known Floodlight controller. We present latency and throughput results of OpenDaylight SDN Controller and Floodlight under different scenarios. We find that OpenDaylight SDN Controller has low average responses as compared to Floodlight. We also note that the standard benchmarking tool - *Cbench* - has no support for real traffic patterns in a data centre, since data centre traffic is considerably complex. In addition to benchmarking of OpenDaylight SDN Controller, we propose modifications in *Cbench* to accommodate models of real traffic in data centres. We also discuss our initial implementation.**

***Keywords*-Software Defined Networking, Benchmarking, Cbench, OpenDaylight, OpenFlow.**

I. Introduction

Enourmous amount of data has resulted into large data centres. Availability of large compute and storage through these data centres has further resulted into new business models. Increase in the complexity of compute and storage has also resulted into more complex computer networks. In a data centre, virtual networks are now common along with virtual machines. Managing networking in data centres is a daunting task which requires innovation. As a result, Software Defined Networking (SDN) has emerged.

Software Defined Networking (SDN) advocates separation of control and data plane. This paradigm shift in networking architecture has the promise of resolving several problems in large scale networks, data centers in particular.

An SDN controller has complete view of the network. The controller also has the ability to change the network structure and services at run time. Several SDN controllers have been developed and deployed. As of now, almost all the SDN controllers are based on OpenFlow protocol such as NOX [9] and Beacon [5].

One of the basic ideas of SDN is to avoid vendor locking. However, dependance on one protocol - OpenFlow - is not serving the purpose. Prior studies have also emaphasied that OpenFlow's current design cannot meet the needs of high-performance networks [11]. Recently - in April 2013 - OpenDaylight consortium is launched with an objective of standardizing SDN development. OpenDaylight SDN Controller (ODL) presents a new SDN controller architecture based on Services Abstraction Layer (SAL) concept such that it supports protocols other than OpenFlow.

SDN controllers are primarily deployed in large scale networks where performance is a critical issue. Controller benchmarking is a commonly used procedure for the performance analysis of SDN controllers. Several studies have been performed regarding benchmarking of commonly available SDN controllers [6], [3], [4]. However, to the best of our knowledge, no such study is available which covers OpenDaylight SDN Controller.

We have performed performance analysis of ODL and Floodlight using *Cbench*. In this paper, we present initial results in the benchmarking of OpenDaylight controller using latency and througput mode in *Cbench*. Initial experiments suggest that OpenDaylight SDN Controller (ODL) is not mature enough to be deployed in the industry. [1]

Cbench [6] is the most commonly used tool for SDN controller benchmarking. It is a stress testing tool which operates in either latency or in throughput mode. *Cbench* is also a blind tool. If multiple instances of *Cbench* are connected with a single controller, *Cbench* instances are unaware of each other.

Since SDN controllers are likely to be deployed in a data centre, it is important that these controllers are tested according to the potential data centre traffic. Data centre traffic is different from other networks because of several reasons. Data centres may have large East-West traffic along with Nort-South traffic. For instance, virtual machines migrations occur frequently. Characteristics of data centre traffic are captured in several studies and models of datacentre traffic already exist [1], [2]. However, there is no tool which makes use of these models for testing and benchmarking SDN controllers.

In this paper, we also propose modifications to *Cbench* such that it also supports stochastic models of traffic for data centers. We also present our initial prototype which is capable

[1]By the time this work is submitted, OpenDaylight has released a new version -*Hydrogen* - not considered in this study.

978-1-4799-7615-7/14/$31.00 ©2014 IEEE

TABLE I
SDN CONTROLLERS

Controller	Language	Created by
Nox	C++	Niciria Networks
Maestro	Java	Stanford University
Beacon	Java	Rice University
Floodlight	Java	Big Switch Networks
Trema	Ruby and C	NEC
Node.Flow	JavaScript	DreamersLab
OpenDaylight	Java	Cisco and OpenDaylight

of using simple probabalistic traffic model.

Our work so far suggests that there is lot of room for improvement in ODL since its performance is not yet comparable with other existing SDN controllers.

This paper is organized such that the next section describes the related work. Section III briefly discusses OpenDaylight SDN Controller. Section IV describes initial results from ODL benchmarking. Section V describes proposed changes to *Cbench* and Section VI outlines future work.

II. RELATED WORK

Increasing interest in Software Defined Networks has lead to the development of many SDN controllers. While most of these SDN Controllers are developed in an academic setting, some of them have gained intesrest of industry as well such as NOX [9], Beacon [5] and Floodlight. A list of common SDN Controllers can be seen in Table I, which is a modified version of Table 5 presnted in [7].

With the clear vision that these controllers are to be deployed in large and scalable networks - such as data centers - various benchmarking studies have been performed to date [3], [6]. These studies have resulted in greater insight of SDN controllers. In an early benchmarking study [6], several controllers - NOX, Beacon, and Maestro - are analyzed comparatively by Amin Tootoonchian. According to the study, NOX can control around 30K flow initiations per/sec. This rate is extremely low from the median rate of 100K flows/sec on a cluster of 1500 servers [8]. The authors of the paper developed an improved version of NOX known as NOX-MT which resulted in to better performance.

Similarly, in another study [3], various controllers are tested for performance including Floodlight, Beacon, NOX-MT and Maestro. In this study, the controllers are evaluated with respect to their architecture. Authors concluded that the performance of controllers varies on the basis of architectural features like switch partitioning, packet batching and multi core support. They found that the controllers which implement static switch partitioning and static batching, have high throughput performance. While the controllers which implement adaptive batching technique, have good latency performances.

It must be however noted that all the controllers discussed in the literature are OpenFlow [12] based controllers. While the inventions and development in this field are on going, reliance on one SDN protocol has slowed the progress and hit the major objective of removing vendor locking altogether. As a result, efforts are being made to develop SDN controllers which support protocols other than OpenFlow. OpenDaylight is a consortium created with the help of major industry players to achieve the same objective. Cisco has already contributed an open source SDN controller to the ODL community known as OpenDaylight Controller. OpenDaylight SDN Controller (ODL) presents a new SDN controller architecture based on Services Abstraction Layer (SAL) concept. ODL also supports protocols other than OpenFlow. The work presented in this paper differes from all previous studies since no previous study has considered OpenDaylight SDN Controller.

SDN controllers are to be deployed in large data center networks. These networks are different from the traditional networks. To understand the traffic models of such networks, various studies have been conducted. For example, Christina Delimitrou [2] differentiates data center network traffic from traditional network traffic. User traffic patterns are different spacially and temporally. The authors present a modeling scheme that has the features of scalability and accuracy. The scheme known as ECHO contains distribution fitting and Markov chain models. These models have different features and can be used to generate traffic models which can capture the characteristics depending on space and time. Although there are some efforts toward more flexible benchmarking tool such as in [10] by Michael Jarschel. However, no traffic models are incorporated in any benchmarking tool. We also present modifications to standard benchmarking tool - *Cbench* - such that it also supports probabilistic models of data center traffic.

III. OPENDAYLIGHT OVERVIEW

OpenDaylight is an open source project supported by IBM, Cisco, Juniper, VMWare and several other major networking vendors [13]. OpenDaylight is an SDN controller platform implemented in Java. As such, it can be deployed on any hardware and operating system platform that supports Java. The architecture of OpenDaylight SDN Controller is shown in Figure 1.

OpenDaylight SDN Controller has several layers. The top layer consists of business and network logic applications.The middle layer is the framework layer and the bottom layer consists of physical and virtual devices.

The middle layer is the framework in which the SDN abstractions can manifest. This layer hosts north-bound and south-bound APIs. The controller exposes open northbound APIs which are used by applications. OpenDaylight supports the OSGi framework and bidirectional REST for the northbound API. The business logic resides in the applications above the middle layer. These applications use the controller to gather network intelligence, run algorithms to perform analytics, and then use the controller to orchestrate the new rules, if any, throughout the network.

ODL is created with an objective of reducing vendor, locking therefore it supports protocols other then OpenFlow. The southbound interface is capable of supporting multiple protocols such as OpenFlow and BGP-LS as separate plugins.

The SAL determines how to fulfill the requested service irrespective of the underlying protocol used between the controller and the network devices.

IV. Performance Evaluation

In this section, we provide detailed performance analysis of Floodlight and ODL. We first describe evaluation testbed and then present latency and throughput results. In the last part of this section, we briefly discuss our observations.

A. Experiment Setup

In order to test SDN controllers, we used a total of 5 PCs (HP Compaq 6300 Desktop PC) all having Intel Core i5 processor. All the PCs were connected with each other through a switch. One of these PCs was dedicated to run the controller and on all others, we ran *Cbench* which is the standard tool used for evaluating SDN OpenFlow Controllers. This configuration is shown in Figure 2. The PC running the controller was provided with 8 GB of main memory and for the others, 4 GB was provided.

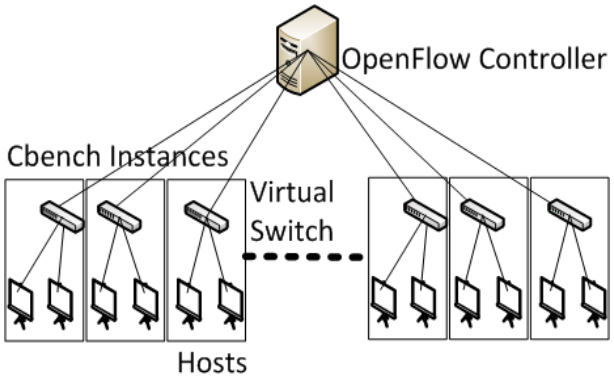

Fig. 2. Testbed Setting

We used different PCs for *Cbench* instances and the Controller in order to reduce the interference and also it would better stress the controllers to be tested as all the instances would be sending *packet Ins* at the same time. We tested the controllers throughput and latency with different number of switches. We ran the controllers with typical learning switch application. The learning switch application performs MAC address learning by storing the output port against each MAC address. Then for each *packet In*, it finds the destination port by looking up the map and sends the packet on that port. Otherwise, the packets are flooded. In all previous benchmarking studies, learning switch application is used [6], [3], [4], as it has reasonable memory requirements for a defined number of switches. This application allows to test the I/O throughput or latency of the controllers.

We tested the controllers with 8, 16 and 32 switches, each switch having 100,000 unique source MACs. For the 8 and 16 switch scenario, each *Cbench* instance emulated only one switch. Each *Cbench* instance was assigned to a unique core. For the 32 switch scenario, we maintained the condition of assigning a *Cbench* instance to unique core. However, each *Cbench* instance now emulated two virtual switches.

We conducted these experiments for three different types of controllers: OpenDaylight SDN Controller Platform (OSCP), OpenDaylight Controller and Floodlight. OSCP was initially supported by OpenDaylight, but that support was dropped later in the favour of ODL. ODL is based on Service Abstraction Layer(SAL) architecture and also supports north bound protocols other than OpenFlow. However, our tests are based on OpenFlow plugin only, provided with ODL. Here, we present results for ODL and Floodlight only. Floodlight is chosen because it is a mature SDN controller and studied in depth in the past.

The SDN controllers can be run in two modes, reactive where the flow setup is performed dynamically for each new flow, or proactive where the flow setup is done statically before the packet arrival. The setup we used was for the reactive forwarding. The controllers were tested in different scenarios and JVM configurations. The results of these tests and the behavior of the controllers are described in next two subsections.

B. Latency Tests

Latency of the controller means that how much time it takes to process a single packet. We tested the controller for latency performance. For this purpose, we ran *Cbench* in latency mode which sends a *packet In* to the controller and waits for response before sending another packet. We evaluated the controllers with different number of virtual OpenFlow switches emulated by *Cbench*. Each latency test was run for a minimum of 10 minutes and repeated 10 times.

From the results, we observed that Floodlight controller provides results pretty much consistent with previously conducted studies. With 8 switches, Floodlight produces 1214 responses/sec per switch on average. Increasing the number of switches resulted into slight increase in the number of responses, from 1214 with 8 switches to 1239 with 16 switches, and to 1335 with 32 switches.

These results are internally consistent as well. For all the evaluated configurations, the minimum number of responses per second per switch is recorded to be 1208 whereas the maximum is recorded to be 1404. However, we noted that variability in tests results increases with an increase in number of switches. The average standard deviation per switch is recorded to be 3.22, 7.74 and 8.51 for 8, 16 and 32 switch scenarios respectively. Floodlight latency test results are summarized in Figure 3.

As compared to Floodlight, the numbers of responses on ODL were unexpectedly very low. The responses recorded were 55/sec, 27/sec and 15/sec on average with 8, 16 and 32 switches respectively for a single switch in the network. We also noticed that while the latency performance of Floodlight increases when the number of switches are increased but ODL shows unexpected behavior as the numbers of responses decrease when the number of switches are increased. ODL latency mode results are summarized in Figure 4.

We also noticed that for ODL, *Cbench* failed to produce

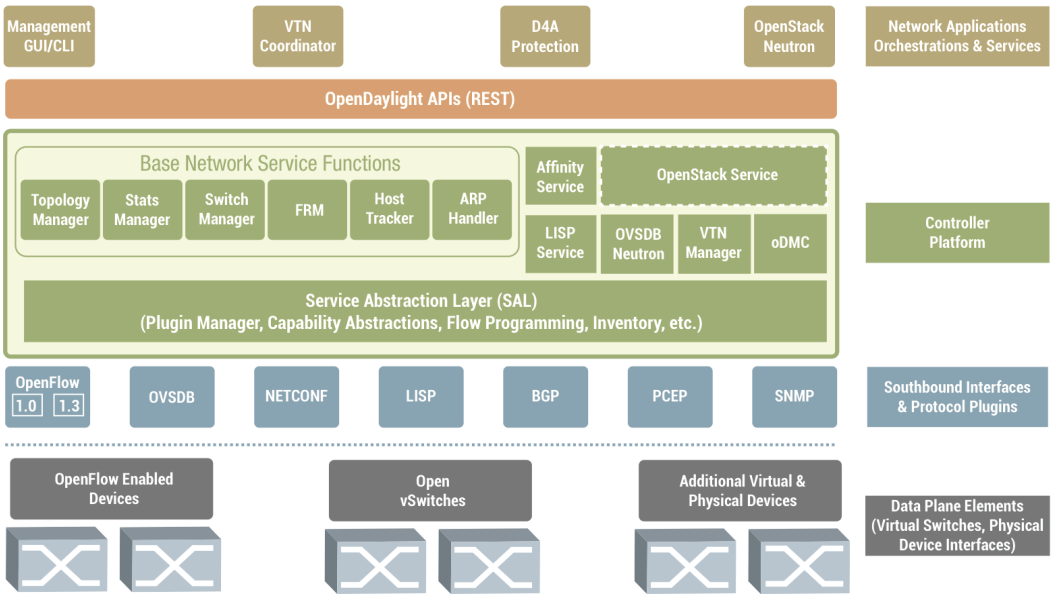

Fig. 1. OpenDayligyt SDN Controller architecture [13].

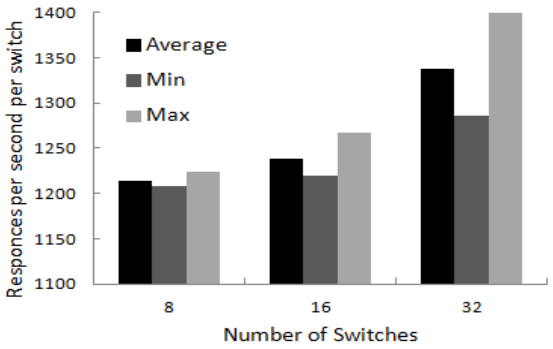

Fig. 3. Floodlight Latency Results

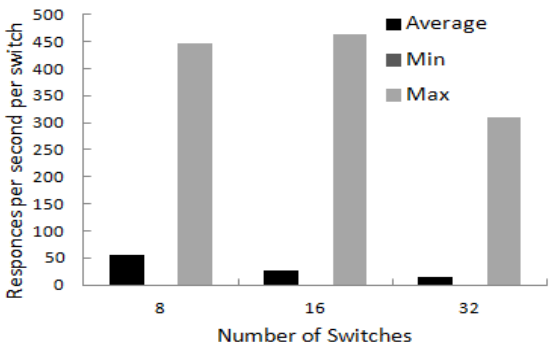

Fig. 4. ODL Latency Results

any result in several tests and showed zero responses. The frequency of such incidents increased with an increase in number of switches - as shown in Table II. Such events are non existent in Floodlight latency tests. If we exclude such tests from our analysis, the number of responses is still very low in ODL.

TABLE II
FAILED LATENCY TESTS IN ODL

Switches	Total Tests	Failed Tests
8	80	20%
16	160	52%
32	320	58%

C. Throughput Tests

In throughput mode tests, the controller is evaluated for how many packets it can process in a second. *Cbench* can also be used to test the controller for throughput performance by running it in throughput mode in which it sends a large control traffic and then records the number of responses for the requests it has sent to the controller. We tested the controller throughput performance with respect to switch scalability. Testing scenarios were same as latency tests except for the fact that tests were run for a minimum of 60 seconds and each test was repeated 6 times.

Floodlight's average responses/sec with 8 switches gives a good figure of 18001. As we increased the number of switches to 16 and 32, the number of responses per second decreased to 9408 and 4536 respectively. Total number of responses to all *Cbench* instances remained almost unchanged with an average of 144006, 150531 and 145155 for 8, 16 and 32 switches respectively. A summary of Floodlight throughput results is shown in Figure 5.

The ODL had again provided very low responses as compared to Floodlight. Average per switch responses recorded on ODL

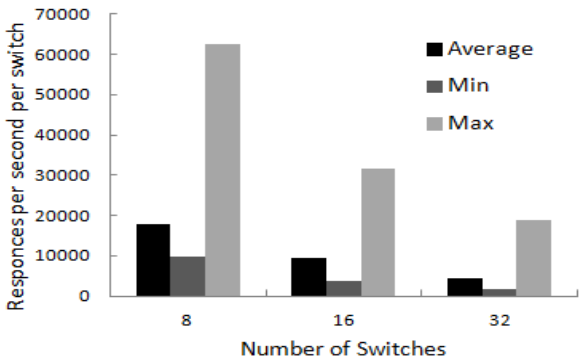

Fig. 5. Floodlight Throughput Results

controller were 270, 33 and 34 for 8, 16 and 32 switches respectively. Total number of responces to all the *Cbench* instances for a given test are also fairly low and variation between tests is also high. These results are shown in Figure 6 and possible reasons are discussed in next subsection.

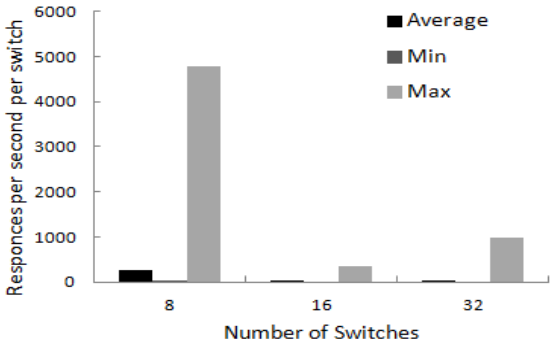

Fig. 6. ODL Throughput Results

D. Discussion

It is evident from results that ODL has several performance issues. We initially found that for each *packet In*, ODL generated a barrier request. However, removal of this behavior did not help much. We also noticed that many ODL tests failed to produce any result. Tabel II describes percentage of failed tests for ODL in latency mode. It is evident that large number of tests are failing and number of failed tests is increasing with number of switches.

We believe that ODL may have memory leaks which can be found out with the help of code profiling. Another possible reason may be that the OpenFlow plugin used with ODL is malfunctioning.

V. *Cbench* AND DATACENTER TRAFFIC

Cbench is a stress testing tool used to evaluate OpenFlow based SDN controllers. *Cbench* tests run in either latency or in throughput mode. In latency mode, *Cbench* sends a *packet Ins* message to the controller and waits for the response. In throughput mode, *Cbench* sends a stream of *packet In* messages to controller for specified period of time. In both cases, *Cbench* counts number of responses from controller. In a real datacentre, traffic patterns are not that simple. It has been shown in several studies that complex probabilistic models can predict datacenter traffic [1], [2]. *Cbench* has no support for such models.

Contemporary studies run *Cbench* tests for a short duration - in the order of tens of seconds for throughput tests. It is also desireable that tests mimicing real datacenter traffic are run for a longer period of time - in the order of days. This is not possible in current *Cbench*.

We propose an enhancement in *Cbench* such that it can create custom traffic patterns and runs tests according to the custom traffic plan. This requires modifications in existing *Cbench* and a design of an additional software layer.

One of the fundamental reason that *Cbench* does not support custom traffic is that all *Cbench* instances connected to the controller, are unaware of each other. As a result, an individual *Cbench* instance works in oblivion and cannot coordinate with other *Cbench* instances. A testing plan based on complete data center requires the knowledge of complete topology.

We propose to obtain the knowledge of complete data center by implementing an additional software layer to *Cbench*. As a result the benchmarking testbed may change as shown in Figure 7. The knowledge of complete data center topology can help mimic connections between actual hosts in the data center.

The jobs of the proposed software layer are: 1) to obtain knowledge of all *Cbench* instances connected to a controller, 2) create custom traffic plans using complete knowledge of topology, 3) to communicate these plans to *Cbench* instances, and 4) to gather statistics. In currnt *Cbench* implementation, it is not possible to exert variable load from different hosts and switches. In the presece of additional proposed software layer, this can be achieved.

The proposed Software Layer is shown in Figure 7. We have created a prototype implementation of software layer and modified *Cbench* to support this prototype. The modified *Cbench* is capable of connecting with software layer, receiving traffic plan, executing tests as per traffic plan and to forward statistics to the software layer.

As of now, the software layer supports two different types of traffic plans. The first model is used to create a traffic plan for the network where it will coin a toss for each switch each time and then decide the switch to communicate with the controller. This involves simple user provided probability, whether a switch will send traffic to the controller at a particular time or not.

The second model considers that communication within the datacenter will require flow establishment between two hosts. For instance, vitual machine migration from one hardware machine to another may require setting up of flows on networking devices. This action may require setting up flow information

on more than one switches at the same time. We simulate this scenario by incorporating a probabilistic model that allows to create random connection between two switches at each instance of time.

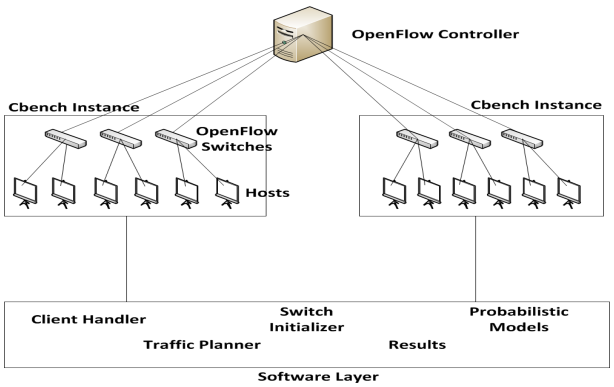

Fig. 7. Proposed Architecture

Once the traffic plan is prepared, it is communicated to all *Cbench* instances. Now *Cbench* instances send *packet In* according to this traffic plan. Additionally, after completing the test, all *Cbench* instances send results to the software layer. The software layer can now derive summarized results of whole topology which is not possible in standard *Cbench*.

VI. Conclusion and Future Work

We have presented results of ODL and Floodlight SDN controllers benchmarking. We have found that benchmarking of ODL with *Cbench* is not very successful. It appears that ODL has several problems such as memory leakages. To pin point these problems and to improve the ODL, we plan to perform code profiling of ODL.
We have also presented an enhanced *Cbench* and its prototype implementation. We plan to incorporate more data center traffic models in enhanced *Cbench*.

Acknowledgment

The authors would like to thank Syed Ali Khayam for useful discussions in the early phase of this study.

References

[1] Theophilus Benson , Ashok Anand, Aditya Akella, and Ming Zhang. “Understanding data center traffic characteristics.” ACM SIGCOMM Computer Communication Review 40, no. 1 (2010): 92-99.

[2] Christina Delimitrou, Sriram Sankar, Aman Kansal, and Christos Kozyrakis. “ECHO: Recreating network traffic maps for datacenters with tens of thousands of servers.” In Workload Characterization (IISWC), 2012 IEEE International Symposium on, pp. 14-24. IEEE, 2012.

[3] Syed Abdullah Shah, Jannet Faiz, Maham Farooq, Aamir Shafi, and Syed Akbar Mehdi. ”An architectural evaluation of SDN controllers.” In Communications (ICC), 2013 IEEE International Conference on, pp. 3504-3508. IEEE, 2013.

[4] Marcial Fernandez. “Evaluating OpenFlow Controller Paradigms.” In ICN 2013, The Twelfth International Conference on Networks, pp. 151-157. 2013.

[5] David Erickson. “The beacon openflow controller.” In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, pp. 13-18. ACM, 2013.

[6] Amin Tootoonchian, Sergey Gorbunov, Yashar Ganjali, Martin Casado, and Rob Sherwood. “On controller performance in software-defined networks.” In USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE). 2012.

[7] Adrian Lara, Anisha Kolasani, and Byrav Ramamurthy. “Network innovation using openflow: A survey.” IEEE Communications Surveys and Tutorials , Vol 16 No 1 (2013) pp 1-20.

[8] Sirikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronie Chaiken, “The nature of data center traffic: measurements and analysis.” In Proceedings of IMC 2009 (2009), ACM, pp. 202208.

[9] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martn Casado, Nick McKeown, and Scott Shenker.“NOX: towards an operating system for networks.” ACM SIGCOMM Computer Communication Review 38, no. 3 (2008): 105-110.

[10] Michael Jarschel, Frank Lehrieder, Zsolt Magyari, and Rastin Pries. “A Flexible OpenFlow-Controller Benchmark.” In Software Defined Networking (EWSDN), 2012 European Workshop on, pp. 48-53. IEEE, 2012.

[11] Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. “Devoflow: scaling flow management for high-performance networks.” In ACM SIGCOMM Computer Communication Review, vol. 41, no. 4, pp. 254-265. ACM, 2011.

[12] OpenFlow Consortium. OpenFlow Specifications. http://www.openflowswitch.org.

[13] OpenDaylight Consortium. http://www.opendaylight.org.