PERFORMANCE EVALUATION OF *SDN* CONTROLLERS: FLOODLIGHT AND OPENDAYLIGHT

SHIVA ROWSHANRAD*, VAJIHE ABDI AND MANIJEH KESHTGARI

Department of Computer and Information Technology, Shiraz University of Technology, Shiraz, Iran.

corresponding author: shiva.rrad@gmail.com

(Received: 27th Jan. 2016; Accepted: 25th Apr. 2016; Published on-line: 30th Nov. 2016)

ABSTRACT: Software Defined Networking (SDN) is a new network architecture. One of its components is the controller, which is the intelligent part of SDN. Many controllers such as Floodlight, OpenDaylight, Maestro, NOX, POX, and others have been released. The question is which controller can perform better in which situations. Many works have been done to compare these controllers with respect to architecture, efficiency, and controllers' features. In this paper, an evaluation based on some network QoS parameters is done. Two of the most popular controllers, Floodlight and OpenDaylight are compared in terms of delay and loss in different topologies and network loads. This paper can help researchers to choose the best controller in different use cases such as clouds and multimedia. The results, with a 95% confidence interval, show that OpenDaylight outperforms Floodlight in low loaded networks and also for tree topologies in mid-loaded networks in terms of latency. Floodlight can outperform OpenDaylight in heavily loaded networks for tree topologies in terms of packet loss, and in linear topologies in terms of latency. There is no significant difference in performance of Floodlight and OpenDaylight controllers in other cases.

ABSTRAK: Software Defined Network (SDN) adalah seni bina rangkaian baru. Salah satu komponennya adalah pengawal, yang merupakan bahagian pintar SDN. Banyak pengawal seperti Floodlight, Open Daylight, Maestro, NOX, POX dan lain-lain telah dikeluarkan. Persoalannya ialah pengawal yang mana boleh berfungsi dengan lebih baik di dalam situasi yang mana. Banyak kajian telah dibuat untuk membandingkan pengawal mengenai seni bina, kecekapan dan ciri-ciri pengawal. Tetapi dalam kertas kerja ini, penilaian berdasarkan beberapa parameter rangkaian QoS dilakukan. Dua pengawal yang paling popular, Floodlight dan OpenDaylight dibandingkan dari segi kelewatan dan kehilangan di dalam topologi dan muatan rangkaian yang berbeza. Kertas kerja ini boleh membantu para penyelidik untuk memilih pengawal yang terbaik dalam kes penggunaan yang berbeza seperti awan dan multimedia. Keputusan mempunyai 95% interval keyakinan menunjukkan bahawa prestasi OpenDaylight melebihi pencapaian Floodlight dalam rangkaian muatan rendah dan juga untuk topologi pokok dalam rangkaian muatan pertengahan dari segi kependaman. Floodlight boleh mengatasi prestasi OpenDaylight dalam rangkaian muatan berat untuk topologi pokok dari segi kehilangan paket dan dalam topologi linear dari segi kependaman. Tidak ada perbezaan yang signifikan dalam prestasi pengawal Floodlight dan OpenDaylight dalam kes-kes lain.

KEYWORDS: *floodlight; mininet; opendaylight; SDN*

1. INTRODUCTION

The idea of programmable networks was introduced to meet the challenges that were faced by traditional networks and to facilitate network evolution. As a result, Software Defined Networking (SDN) is a new paradigm that revolutionized traditional network architecture [1]. Actually, SDN is a framework for automatically controlling large devices, services, topologies, traffic paths, policies, and APIs in a network [1]. The main idea of SDN is related to the physical separation of two parts of the networks: the control plane and data plane. The control plane is separated from the data plane and is moved to a centralized server called the controller. This decoupling abstracts infrastructure from applications and causes flexibility, programmability, cost efficiency, and novel network architecture. In this case, the main role of a controller is to set up rules or make decisions for packets and the main role of data plane is to direct or forward packets based on the rules coming from controller. The controller is the intelligent part of network and the data plane is the slave to the controller's orders. Southbound and northbound interfaces are two other components in a software defined network. A southbound interface connects switches (data plane) to the controller and a northbound interface offers APIs for network controlling and services [2]. One of the most famous implementations of the southbound interface is OpenFlow [3] protocol. According to [3, 4] OpenFlow is an open interface that is used to control the forwarding tables remotely in switches, routers, and access points. Many controllers are developed based on OpenFlow such as POX [5], NOX [6], Beacon [7], OpenDaylight [8], Floodlight [9] and so many others. Choosing the best controller can be problematic for network administrators and researchers. There are existing works to compare controllers in architectural and efficiency aspects such as scalability and availability, and also from the feature parameters viewpoint such as productivity, language, and Interface and platform support. In this paper, two common controllers, Floodlight and OpenDaylight, are compared to each other using Mininet [10], an SDN emulator, in terms of QoS parameters such as delay and loss. The comparison is done for different topologies and network loads. The results can help to choose the best controller for networks that have QoS requirements, such as networks that carry multimedia traffic, or for other special use cases, such as datacenters and clouds.

The rest of this paper is organized as follows: In Section 2 related works are presented. Section 3 introduces the controllers that are compared in this paper. There is also a short introduction to Mininet and its supported topologies in this section. Emulation and analysis are presented in Section 4. The motivation for this, and future, works are discussed in Section 5. Finally, Section 6 concludes the paper.

2. RELATED WORKS

In recent years, many works have been done to compare SDN controllers, a review of some of which is presented in this section. In [11], four open-source OpenFlow controllers including NOX, Beacon, Maestro, and Floodlight, were compared from architectural aspects, on a shared memory multi-core machine. Four key performance bottlenecks are considered, including multi-core support, switch partitioning, packet batching, and task batching. Architectural designs, including static switch partitioning and static batching are employed by Beacon, NOX-MT, and Floodlight, while shared-queue and adaptive-batching are employed by Maestro. The performance evaluation results show that static switch partitioning and packet batching designs are best for controllers in high throughput networks. Packet batching and task batching designs are good for controllers with delay-sensitive control plane applications. The results also show that sending control messages out individually can improve the latency performance. Based on these results, a controller was proposed whose performance was better than the compared controllers.

In [12], five controllers named POX, Ryu, Trema, Floodlight, and OpenDaylight are compared to each other. According to Khondoker et al. [12], choosing a controller is a Multi-Criteria Decision Making (MCDM) problem because several properties of a controller are important for users. The authors chose an Analytic Hierarchy Process (AHP) in management science because of pair-wise prioritization and an integrated consistency checking mechanism [12]. POX, Ryu, Trema, Floodlight, and OpenDaylight were compared based on available interfaces, supporting of virtual switching, GUI, supporting of REST API, productivity, being open source, having documentation, age, modularity, supporting language, platform, TLS, OpenFlow and OpenStack networking. The results show that Ryu got the best value (0.287) in this priority vector. Floodlight, OpenDaylight, Trema, and Pox got the next best results respectively. It is noticeable that the priority vector values of Floodlight and OpenDaylight (0.275, 0.265) were very close to that of Ryu.

Shalimov et al. [13] proposed a method to test and compare different open-source SDN/OpenFlow controllers including NOX, POX, Beacon, Floodlight, Mul, Maestro, and Ryu. Performance metrics including throughput and latency, scalability, reliability, and security were considered for comparison. The test of scalability and performance were done with Cbench. Reliability and security test were performed with hcprobe. In three experiments of finding average throughput with different number of threads, different number of switches, and different number of hosts, Beacon achieved the maximum throughput. For the average response time of controllers with 105 hosts, Mul, Beacon and Floodlight achieved minimum latency. In determining reliability between controllers, all except Mul and Maestro coped with the test load. For security measurements, five tests were performed including incorrect message length, invalid OpenFlow version, incorrect OpenFlow message type, malformed packet-In message, and malformed port status message. Ryu best coped with these four cases across these five tests.

Al-Somaidai and Yahya [14] discussed five different versions of OpenFlow switch standard including 1.0, 1.1, 1.2, 1.3 and 1.4, four different platforms for simulation and emulation of SDN including Mininet, EstiNet, NS-3 and Trema, seven types of controllers including NOX, POX, Floodlight, OpenDaylight, Ryu, Mul and Beacon, and also different switch software and tools. They mentioned floodlight and OpenDaylight as the controllers with good documentation and flexibility.

Kaur et al. [15] used POX controller and Mininet to emulate a SDN and verified the behavior of network applications in POX. Also, the top five controllers including POX, Ryu, Trema, Floodlight, and OpenDaylight were compared to each other with six features including language support, OpenFlow support, being open-source, having a GUI and REST API, and also platform support.

Govindraj et al. [16] stated that in the comparison of different switches, OpenFlow based switching provides load balancing and reduces bandwidth wastage (due to use of the loop-free topology), so that this kind of switch has better utilization in data centers and enterprises. They also concluded that the usability of OpenFlow can be increased. Several other comprehensive SDN surveys [17-19] are available.

3. FLOODLIGHT, OPENDAYLIGHT AND MININET

As presented in section 2, many works have been done to compare SDN controllers. Among the controllers, Floodlight and OpenDaylight were two controllers with closely similar results. Also recently, these controllers were the focus of more research and papers than other controllers, especially in Cloud computing, QoS, and Multimedia fields. The performance of these controllers were compared in terms of network QoS parameters using Mininet. In this section, Floodlight and OpenDaylight are reviewed and also a short introduction to Mininet is presented.

3.1 Floodlight

This controller is based on the Beacon controller from Stanford University and works with physical and virtual OpenFlow switches. Some of the features of this controller are as follows: apache-licensed, Java-based (non-OSGI), modular, event-driven, asynchronous application framework, thread-based, and using a synchronized lock. Components of this controller are topology management, use for discovery of both OpenFlow and non-OpenFlow endpoints (LLDP), device management including MAC and IP tracking, path computation, infrastructure for web access which is used for management, counter store used for OpenFlow, BigDB (NoSQL, Cassandra-based database) and Quantum plug-in that causes interoperation with element agents supporting OpenFlow. These components are loadable services; and Floodlight Provider is the core module that handles inputs and outputs receives from and sends to switches. This module also translates OpenFlow messages into Floodlight events. There are REST APIs in controller for getting and setting the controller's state, event notification systems, and passing emitted events by Java Event Listeners. Floodlight also has sample applications including learning switch, hub application and static flow pusher, Firewall and load balancer [9].

3.2 OpenDaylight

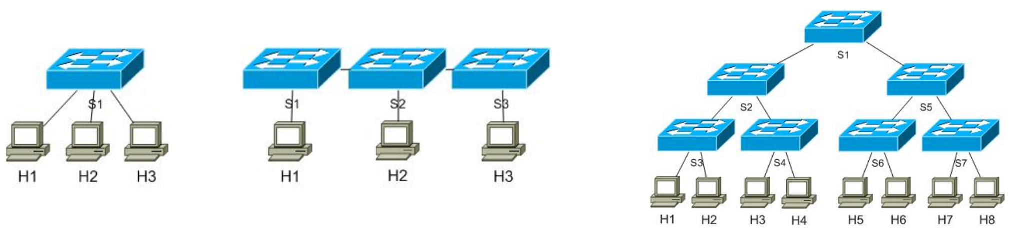
In 2013, the OpenDaylight Project consortium was formed as a Linux Foundation project. This controller creates a de facto northbound API standard so that different southbound protocols, like OpenFlow, I2RS, and NETCONF can be programmed. Some of the features of this controller are as follows: Java-based (OSGI), modular, pluggable and supporting of multiple southbound protocols. OpenDaylight supports the programming of a bidirectional REST and OSGI framework that supports applications running in the same address space as the controller. Internal and external requests for services will be mapped by a service abstraction layer to the appropriate southbound plug-in and a basic service abstraction that higher-level services are built upon, will be provided; this feature depends on the capabilities of the plug-ins. Topology abstraction and discovery, PCE-P (and CSPF), OpenFlow, I2RS (as it evolves), and NETCONF are other built-in services within in OpenDaylight [8]. According to [8, 9, 12] a comparison of these two controllers is done and shown in Table 1.

3.3 Mininet

Mininet is a network emulator, which has been used by developers, teachers, and researchers to create virtual networks and SDN implementation. It was developed by the Mininet Team to run controllers, switches, and hosts in a virtual network on a single machine. The default topology in Mininet consists of an OpenFlow kernel switch connected to two hosts and an OpenFlow controller. Hosts on Mininet are able to run underlying Linux and file system commands [10]. For example, "ipref" parses bandwidth between a client and server and "topo" makes topologies with the Python API for custom virtual networks. There are also three predefined common topologies in Mininet which are

Table 1: A comparison of Floodlight and OpenDaylight [8, 9, 12]

Feature	Floodlight	OpenDaylight
Developer	Big Switch Networks	Linux Foundation
Supporters	Big Switch Networks	Cisco, HP, IBM , Juniper, VMWare, etc.
Written language	Java	Java
Supporting language	Java, Python and any language supports Rest API	Java
REST API	Yes	Yes
OpenStack networking (Quantum)	Yes	Yes
TLS supporting	Yes	Yes
Open-source	Yes	Yes
OF version	Full support for 1.0 and 1.3, experimental support for OpenFlow 1.1 and 1.2	1.0 , 1.3
User interface	web, Java	Web
Interfaces	southbound (OpenFlow), northbound (Java, REST)	southbound (OpenFlow and other SB protocols), northbound (Java RPC, REST)
Virtualization	Mininet, OpenVswitch	Mininet, OpenVswitch
Platform	Linux, Mac, Windows	Linux, Windows
Active community	Yes	Yes
Age	4 years	3 years
Documentation	Good (documentations exist in official website or other developers sites)	Medium
Mailing list activity	Very high	Medium
Handling mixed none-OpenFlow and OpenFlow networks	Yes	Yes
Installation	Very easy	Easy
Loop supporting	Topologies	Topologies and OF islands



(a) Single topology (b) Linear topology (c) Tree topology

Fig. 1: Three common topologies in Mininet.

presented in Fig. 1: single, linear, and tree. A single topology has 1 controller, 1 switch, and a definable number of hosts. A linear topology has serial connections with definable number of switches and hosts. A tree topology consists of multiple topology levels with a definable number of levels (depth and fan-out). Remote controllers also can be used in

Mininet. Therefore, a virtual network will be connecting to any remote controller in VM, local machine, or anywhere else. Some other advantages of using Mininet are system-level regression test supporting, complex topology testing and CLI which is topology-aware and OpenFlow-aware. Developed code for an OpenFlow controller, modified switch, or host on Mininet, can move to a real system with minimal deployment. So a network design in Mininet can usually move directly to hardware switches for line-rate packet forwarding [10].

4. PERFORMANCE EVALUATION

All network instances were installed and implemented on Ubuntu 14.0.4 with 4 GB RAM. Native installation from Source on Ubuntu is considered for Mininet installation. Floodlight and OpenDaylight both supports OpenFlow version 1.3. OpenVswitch 2.0.1 with link status of up to 10 Mbps FDX is used as OpenFlow switches. Number of switches and hosts considered for each topology is shown in Table 2. Three different scenarios for each topology are considered. In the first scenario, no cross traffic was generated; while in the second scenario, nearly half of the bandwidth was cross traffic; and in third scenario, almost the whole bandwidth was filled with cross traffic. Each scenario was repeated 10 times. To have a near-real experience, the production of some cross traffic was based on different rates of YouTube video transmission for H264 [20] and Skype video conferencing [21]. Other cross traffic production was on a random basis to simulate any other traffic which can exist in a network.

Table 2: Number of switches and hosts in single, linear and tree topology

Topology	**Number of**	
	Switch	**Host**
Single	1	8
Linear	8	8
Tree (depth=3 , fan-out=2)	7	8

The general network QoS parameters are:

- Latency: Also known as delay, is the amount of time that it takes for a packet to travel along the network from source to destination.
- Jitter: The variation of latency.
- Loss: The percentage of packets that failed to reach their destination. Packet loss ratio is measured in percentage of total packets as in Eq. (1).

$$packet\ loss\ ratio = number\ of\ packets\ lost/number\ of\ packets\ sent \qquad (1)$$

- Throughput: The ability of network to carry a volume of data over a unit of time [22].

Throughput is related to network bandwidth (which is 10 Mbps in all scenarios of this paper) and loss ratio. Jitter can also be calculated using latency values. Hence, the other two network QoS parameters are considered for evaluating controllers.

Since, each scenario was repeated 10 times (less than 30), so the table should be used for confidence interval calculation. Mean difference ($\bar{x}$) was calculated and then a 95% Confidence Interval (with a confidence level of α=0.05) of mean difference was determined by use of Eq. (2), where (1 - α/2) is 0.97 and the freedom degree (n – 1) is 9;

so based on the table, t[0.97, 9] should be used. If the CI range includes zero, it means that two controllers have the same performance [23].

$$(\bar{x} - t_{[1-\propto/_2;n-1]}\ s/\sqrt{n}\ , \bar{x}+t_{[1-\propto/_2;n-1]}\ s/\sqrt{n}) \qquad (2)$$

Tables 3 to 5 show the latency in each scenario. The exact comparison of these two controllers with 95% confidence intervals, are also shown in the last row of each table. In all topologies, latency of a specific flow between host 1 and host 8 is measured. There is no packet loss in first and second scenarios.

Table 3: Latency with no cross traffic (low load network)

		Single	Linear	Tree
Min (ms)	**Floodlight**	0.079	0.221	0.077
	OpenDaylight	0.078	0.221	0.116
Max (ms)	**Floodlight**	9.593	78.808	9.767
	OpenDaylight	1.824	11.848	2.078
Average (ms)	**Floodlight**	0.476	3.279	0.545
	OpenDaylight	0.1 88	0.749	0.254
CI of Averages Difference		(0.278, 0.298)	(1.83, 3.22)	(0.219, 0.363)

Table 4: Latency with half of the bandwidth cross traffic (mid load network)

		Single	Linear	Tree
Min (ms)	**Floodlight**	0.409	1.8	2.244
	OpenDaylight	0.418	2.013	0.577
Max (ms)	**Floodlight**	15.142	302.562	134.093
	OpenDaylight	10.99	253.266	17.107
Average (ms)	**Floodlight**	2.31	52.778	16.38
	OpenDaylight	2.019	45.831	3.656
CI of Averages Difference		(-0.393, 0.975)	(-3.246, 17.139)	(10.124, 15.32)

Table 5: Latency with almost full bandwidth cross traffic (high load network)

		Single	Linear	Tree
Min (ms)	**Floodlight**	1171.316	1143.583	1065.705
	OpenDaylight	1165.575	1159.357	1205.207
Max (ms)	**Floodlight**	1357.046	1488.846	1634.719
	OpenDaylight	1450.293	1995.651	1235.842
Average (ms)	**Floodlight**	1232.739	1306.647	1235.627
	OpenDaylight	1213.287	1623.714	1212.662
CI of Averages Difference		(-7.802, 46.706)	(-409.314, -224.793)	(-16.609, 62.539)

According to Table 3, maximum latencies in all three topologies belong to Floodlight, but there is no significant difference between the results for minimum latency. Maximum latency in Table 3 is related to the first packet received by switch which has no entry in its forwarding table for the flow. It is noticeable that maximum latency in Floodlight is much more than OpenDaylight, it means that Floodlight needs more time to find the route and send a decision for newly arriving flows. As the average latencies and their 95% confidence intervals in Table 3 show, networks that use Floodlight as their controller have more latency while the traffic is low.

Besides the actual function of each controller among a solitude network, Floodlight and OpenDaylight must be responsible against a populated network. A traffic generator is used to generate large traffic (5 Mbps for mid loaded network and about 10 Mbps for high loaded network). Unlike the fact that the maximum latency in Table 3 is related to finding an action for the first packet by the Floodlight and OpenDaylight controllers, the maximum latency measurement in Table 4 and Table 5 is about a time that network traffic is high. According to Table 4, for mid loaded networks, in single and linear topologies, with 95% confidence intervals there is no difference in the latency results for Floodlight and OpenDaylight and both act the same because their CI of average latencies difference, includes zero. In a more complicated topology (tree topology), the network with the OpenDaylight controller has less latency. For the heavily loaded network, according to Table 5, there is no significant difference in single and tree topologies, while in linear topology, Floodlight achieves less latency and acts faster. It is also noticeable that packet loss happened *only* in the heavy loaded network and there is no packet loss in low and mid load networks. The loss in heavy network happened at time when bandwidth was becoming full and caused congestion. The packet loss results are shown in Table 6.

Table 6: Average loss Measurements in high load network

	Single	**Linear**	**Tree**
Floodlight	41.6	41.3	34.4
OpenDaylight	15.8	32.4	47.2
CI of Averages Difference	(15.659, 35.941)	(-5.397, 23.197)	(-25.349, -0.251)

Table 7: Best controller in different situations

		Latency	**Loss (high load network)**
Single	**Low load**	OpenDaylight (2.5 times better)	OpenDaylight (2.6 times better)
	Mid load	Same	
	Heavy load	Same	
Linear	**Low load**	OpenDaylight (4.4 times better)	Same
	Mid load	Same	
	Heavy load	Floodlight (1.2 times better)	
Tree	**Low load**	OpenDaylight (2.1 times better)	Floodlight (1.4 times better)
	Mid load	OpenDaylight (4.5 times better)	
	Heavy load	Same	

5. MOTIVATION AND FUTURE WORKS

Recently SDN has attracted researchers' attention as an emerging network architecture that can resolve today's network issues in different cases such as multimedia

QoS and cloud and datacenter management. Many papers and research for different use cases has been done using different controllers, among them OpenDaylight and Floodlight were the most popular controllers. Despite this research, there is still a question of which controller can perform better in which situations or use cases. According to features listed in Table 1 it is difficult to decide which controller is really better unless one exact feature, such as supporting non-OpenFlow southbound interfaces, is needed. Therefore, in this paper, these two controllers were compared in terms of network parameters like latency and loss. The comparison was done in three kinds of topology and different network loads. As Table 7 shows, the results can be summarized as bellow:

(a) OpenDaylight can perform better in networks with low load for any kinds of topology.

(b) Both controllers act the same while the network load is about 50% of actual bandwidth, except for the tree topology where OpenDaylight outperforms Floodlight.

(c) For networks with heavy traffic, OpenDaylight outperforms Floodlight only in the single topology because of its better performance in terms of loss.

(d) For networks with heavy traffic, Floodlight can have a better performance than OpenDaylight for linear and tree topologies due to less latency and loss.

(e) OpenDaylight is a better choice for single topologies with various network loads.

(f) Floodlight is a better choice for linear topologies with various network loads.

(g) Floodlight is a better choice for tree topologies with heavy network loads and traffic sensitive to loss, such as video and voice.

(h) OpenDaylight is a better choice for tree topologies with various network loads except for traffic that is sensitive to loss.

According to these results, one can choose the best controller based on the network's features. For example, in data centers and clouds, tree topologies are used. According to (g) and (h) results which are about the performance of controllers in tree topology, OpenDaylight can be a better choice if a module/service for improving its performance in terms of loss being added to it. In another case, Floodlight can be the best controller if a QoS module for improving latency is added to it.

For future work, the number of switches each controller can manage at the same time in different network loads will be tested. The future results, alongside this paper's results and other features of the controllers, can help to make an optimized decision for choosing the best controller.

6. CONCLUSION

Floodlight and OpenDaylight are two common controllers among SDN controllers. Each of these two controllers is modular and could be programmed for new network services. They are both open-source with support Java language. There are also some differences between Floodlight and OpenDaylight. Floodlight has a Java-based user interface while OpenDaylight supports non-OpenFlow control protocols and provides an abstraction layer, above south-bound protocols. There are works done to compare controllers in architectural and efficiency aspects such as scalability and availability and also from the feature point of view. In this paper, network latency and packet loss for Floodlight and OpenDaylight are measured in Single, linear and tree topologies, in various traffic loads. The results with 95% confidence interval show that the controllers have a competitive behavior. There are situations where both controllers perform the same. There are situations where networks with OpenDaylight have better latency, such as in tree topology for a network with half of bandwidth traffic, while floodlight can outperform OpenDaylight in terms of packet loss in heavy loaded network in tree topology. Comparing of these two controllers in more complicated topologies and with respect to the number of switches they can manage is considered for future work.

REFERENCES

[1] Azodolmolky S. (2013) Software defined networking with OpenFlow. Packet Publishing Ltd, UK.

[2] Nadeau TD, Gray K. (2013) SDN: Software Defined Networks. O'Reilly Media Inc., USA.

[3] Pfaff B, Lantz B, Heller B. (2012) OpenFlow switch specification version 1.3.0. Open Networking Foundation, http://www-bcf.usc.edu/~minlanyu/teach/csci599-fall12/papers/openflow-spec-v1.3.0.pdf (2012). Accessed September 2015

[4] Open Networking Foundation (2012) Software-Defined Networking: The new norm for networks. ONF White Paper, https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf. Accessed September 2015

[5] Mccauley J. (2014) Pox: A python-based Openflow controller. POX. http://www.noxrepo.org/pox/about-pox, Accessed: September 2015

[6] Gude V, Koponen T, Pettit J, Pfaff B, Casado M, McKeown N, et al. (2008) NOX: towards an operating system for networks. ACM SIGCOMM Comp. Comm. Rev., 38(3):105-110.

[7] Erickson D. (2013) The Beacon Openflow Controller. Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, 13-18.

[8] OpenDaylight: Open Source Programmable Networking Platform (2015) OpenDaylight, a Linux Foundation Collaborative Project, http://www.opendaylight.org/software. Accessed: September 2015.

[9] Floodlight is an Open SDN Controller. (2015) Project Floodlight. http://www.projectfloodlight.org/Floodlight. Accessed: September 2015.

[10] Mininet an instant virtual network on your laptop (or other PC) (2015). Mininet Team. http://Mininet.org. Accessed: September 2015.

[11] Shah SA, Faiz J, Farooq M, Shafi A, Mehdi SA. (2013) An architectural evaluation of SDN controllers. Communications (ICC) 2013 IEEE International Conference, 3504-3508.

[12] Khondoker R, Zaalouk A, Marx R, Bayarou A. (2014) Feature-based comparison and Selection of Software Defined Networking (SDN) Controllers", World Congress on IEEE on Computer Applications and Information Systems (WCCAIS), 1-7

[13] Shalimov A, Zuikov D, Zimarina D, Pashkov V, Smeliansky R. (2013) Advanced study of SDN/OpenFlow controllers. Proceedings of the 9th Central & Eastern European Software Engineering Conference, ACM, 1-6.

[14] Al-Somaidai MB, Yahya E. (2014) Survey of software components to emulate OpenFlow protocol as an SDN implementation. Am. J. Software Engin. Appl., 3(6):74-82.

[15] Kaur S, Singh J, Ghumman NS. (2014) Network programmability using POX controller. http://sbsstc.ac.in/icccs2014/Papers/Paper28.pdf. Accessed: September 2015

[16] Govindraj S, Jayaraman A, Khanna N, Prakash KR. (2012). OpenFlow, load balancing in enterprise networks using Floodlight controller; University of Colorado; http://morse.colorado.edu/~tlen5710/12s. Accessed: September 2015

[17] Kreutz D, Ramos F, Verissimo P, Rothenberg CE, Azodolmolky S, Uhlig S. (2014) Software-Defined Networking: A comprehensive survey. Proceeding of the IEEE, 103(1): 14-76

[18] Mendonca M, Nunes BA, Nguyen X, Obraczka K, Turletti T. (2014) A survey of Software-Defined Networking: Past, present, and future of programmable networks. Comm. Surveys & Tutorials, IEEE, 16(3):1617-1634.

[19] Lara A, Kolasani A, Ramamurthy B. (2014) Network innovation using Openflow: A survey. Comm. Surveys & Tutorials, IEEE, 16(1): 493-512.

[20] YouTube advanced encoding settings. (2013) Available online: https://support.google.com/youtube/answer/1722171?hl=en, Accessed: September 2015

[21] How much bandwidth does skype needs? (2015) Available online: https://support.skype.com/en/faq/FA1417/how-much-bandwidth-does-skype-need, Accessed: September 2015

[22] De Gouveia F, Magedanz T. (2002) Quality of service in telecommunication network. Telecommunication System and Tchnologies, Encyclopeida of Life Support Systems(EOLSS).

[23] Jain R. (2008) The art of computer systems performance analysis. John Wiley & Sons, Littleton, Massachusetts.