The Nucleus ***54***, *No. 1 (2017) 16-22*

www.thenucleuspak.org.pk

The Nucleus

ISSN 0029-5698 (Print)
ISSN 2306-6539 (Online)

Performance Evaluation of Software Defined Networking vs. Traditional Networks

M.I. Lali[1,*], R.U. Mustafa[2], F. Ahsan[3], M.S. Nawaz[4] and W. Aslam[5]

[1]*Department of Computer Science, University of Gujrat, Gujrat, Pakistan*

[2]*Department of Computer Science, COMSATS Institute of Information Technology, Sahiwal Campus, Pakistan*

[3]*Department of Computer Science and Engineering, HITEC University, Taxila, Pakistan*

[4]*Department of Information Science, School of Mathematical Sciences, Peking University, Beijing, China*

[5]*Department of Computer Science & IT, The Islamia University of Bahawalpur, Bahawalpur, Pakistan*

ikramullah@uog.edu.pk; razaulmustafa@outlook.com; faraz.ahsan@hitecuni.edu.pk; msaqibnawaz@pku.edu.cn; waqar.aslam@iub.edu.pk

ARTICLE INFO

Article history:
Received: 07 November, 2016
Revised: 27 February, 2017
Accepted: 28 February, 2017

Keywords:
SDN
Traditional network
Packet delivery ratio
Latency
Processing overhead

ABSTRACT

Emerging mega trends in Information and Communication Technology (ICT) and many Internet applications have offered new challenges to future Internet, of which dynamic management and high bandwidth rank high. High bandwidth needed for transporting huge data cannot be realized with traditional networking methods, wherein configurations of devices are carried out manually and capability of network infrastructure is not fully utilized. Software Defined Networking (SDN) is an emerging solution for such problems. In this article, we analyze the performance of SDN using its standard OpenFlow protocol by considering a scenario of medium enterprise data center. Performances of SDN are executed for parameters such as latency, packet delivery ratio and processing overhead in various topologies using simulations carried out in Mininet. Results indicate significant performance improvements of SDNs over traditional networks.

1. Introduction

One of the main objectives of future Internet-based applications is to provide connectivity to people such that network services can be properly utilized. However, emerging trends in mobile, social networks, cloud computing and big data have given new challenges to future Internet, for which high bandwidth, easy accessibility and dynamic management play important roles [1]. Traditional network methods that are based on manual configuration of proprietary devices are complex, error-prone and cannot properly utilize the overall capability of physical network infrastructure when it comes to large data like data centers. Old ways of managing networks are difficult to maintain. Problems like configuration, debugging and adding new devices need lot of human input. Furthermore, to mitigate flexibility problems, researchers have invested in an initiative that implements networks with greater programming capabilities and reduces the need to replace switching equipment [2]. These requirements lead to the development of new paradigm in networking known as Software Defined Networking (SDN).

SDN is gaining popularity due to its dynamicity, flexibility, adaptability and cost effective architecture, resulting in better outcomes for high bandwidth usage for enormous data accessibility. Nunes et al. [3] has provided a historic review about the idea of network programmability, which eventually led to SDN revolution. SDN is an emerging networking methodology in which the control plane is separated from the data plane, switches in the network act as data forwarding devices and logically centralized servers control network management [4, 5]. SDN is providing solution to problems that are faced in conventional networks and is gaining more acceptances in applications such as cloud and grid computing. SDN can deal with long-drawn-out security requests (which is unrealistic with hard-wired systems) put on the organization's base by new applications, Bring Your Own Device (BYOD) and Virtual Machine (VM) items.

By separating the data plane from the control plane and moving the control plane to a centralized controller, SDN offers a strong capability for the deployment of a wide-range of network policies (such as routing, fault-tolerance and security) along with the ability to implement new network technologies [6]. Additionally, network administration is much more focused in terms of applications and services rather than topologies and data

[*]Corresponding author

management. The developments of Ethane [7] and OpenFlow [8] have brought implementation of SDN closer to reality.

One of the foremost SDN standard is OpenFlow [8], which is an open architecture developed initially to run experiments on heterogeneous networks without affecting real user traffic. OpenFlow Switch Specification [9] establishes rules for communication between the data plane and the control plane and allows control of entire network through user-defined software applications (APIs). The Open Networking Foundation (ONF) [10] brings together about 90 companies and is dedicated to promote, release and adopt OpenFlow Specification. The OpenFlow architecture [8] consists of three main parts: an external controller, an OpenFlow switch and OpenFlow protocol for communication establishment between switch and controller. Standardization is advanced through recently formed Architecture Working Group and new open source association projects, OpenDaylight [11] and Estinet [12].

Previously some research has been done to investigate performance of SDN. Gelberger et al. [2] analyzed the impact of SDN on parameters such as latency and throughput under different workloads. They have also investigated performance penalty for complex and more functional SDN infrastructure. Results confirm inherent performance penalty in SDN. Banjar et al. [13] used INET framework in OMNeT++ for analyzing the effect of location of OpenFlow controllers on the performance of an OpenFlow network. Their results indicate how OpenFlow network performance is affected by the position of the central controller. Sood et al. [14] proposes an analytical model to study the performance of SDN switches. They have used *M*/Geo/1 model to analyze SDN switches, while extensive simulations are done to validate the proposed model. Results indicate the impact of factors such as flow-table size, number of rules, packet arrival rate and position of rules on SDN switch performance.

Numerous surveys and theoretical literature exists to date [1, 15-17], highlighting various concepts of SDN and providing details. However, in this study we aim to provide a comparison of SDN with traditional network for medium level organizations, which either own a private commodity-off-the-shelf data center or intend to own. Based on performance aim, these organizations need to decide whether to opt for SDN or not. Besides, the physical layout of their infrastructure remains the same. In this article, we have analyzed the performance of SDN using its standard OpenFlow protocol. Mininet [18] is used to simulate scenarios varying in topologies using SDN approach. Mininet also provides a platform for network emulation that focuses on OpenFlow architecture. It uses Linux kernels and Python language scripts to form virtual networks of large number of hosts, OpenFlow switches and controllers on a single desktop/laptop system.

This article is organized as follows. Network parameters used for performance evaluation of SDN and traditional networks are discussed in Section 2. Issues related to placement of controller in SDN are also presented in Section 2. Simulation experiments are performed in Section 3, where SDN performance is compared with traditional networks. Conclusions and future directions are given in Section 4.

2. Network Performance Measure

Each network is different both in nature and design; therefore many different ways are used to measure the performance of a network. In this work, performance of both networks is measured in terms of latency, Packet Delivery Ratio and processing overhead.

2.1 Latency

Bandwidth and latency are two key metrics that reflect on network speed. Network systems associated with low latency are ones that encounter small delay, while a high latency is associated to long gaps between arrivals of packets. In general, actual network bandwidth varies over time. Network bottlenecks may induce high latency of packets, thereby resulting in bandwidth decrease. These bottlenecks may be temporary (lasting for few seconds) or persistent [19].

2.2 Packet Delivery Ratio (PDR)

PDR represents the ratio of the data packets delivered successfully to the destination. In other words, ratio of the number of data packets that reached destination to the total number of packets sent. Thus

$$PDR = \frac{total\ number\ \ of\ received\ \ packets}{total\ \ number\ \ of\ sent\ packets}. \tag{1}$$

2.3 CPU Utilization

The switch CPU performs two different functions when it finishes the boot transform process. These two functions are:

- Switch runs the diverse framework procedures needed for a switch working in a system.
- Switch sends and gets packets to and from the switch hardware.

CPU usage increases when additional time is allowed due to switching or when more packets are sent and received. Under typical working conditions, on a non-stackable switch, the CPU is occupied no less than 5 percent of the time. In the event that the switch is stacked, the CPU is occupied at least 7% of the time. In a switch stack, CPU usage is generally measured by the master switch. The total number of ports in the stack influences the overall CPU use.

In SDN, packet usually moves first on OpenFlow switch, where header field is matched with flow table that is already entered in flow table entries. If current packet entry is found in flow table, it is forwarded to specific port. Otherwise, controller is invoked to decide about the packet, which then informs the switch to drop it or create a new entry in the flow table in order to support new network flows. Controller's main functionality is to establish flows in networks. If the aim is for very low downtime then another controller must be introduced in the network to keep the downtime low [16].

2.4 Packet Switching

Switching and routing of packets are the main functions of a network. Packet switching is most widely used for protocols such as TCP/IP and Ethernet, which are based on this technology. Circuit-switching is used in telephone service, where two parties use a dedicated line for transmission [20].

For robustness, switching and routing designs are traditionally based on distributed approaches. However, such distributed designs have many limitations that include slow convergence rate, complex implementation and limited ability to achieve adaptive control. To overcome these limitations, SDN allows applications to adaptively control a network, applications feeding with status information of global network and offers closed loop control. In order to achieve this in SDN, several ideas are proposed for utilizing SDN platform for better routing designs. Load balancing and cross-layer design are two popular solutions in this regard [1]:

- **Load Balancing:** It is a widely used technique for achieving better resource usage. Front end load balancers are deployed in data centers for directing each request of clients to a particular server replica in order to avoid overloading of the network, reduce response time and increase throughput. However, dedicated load balancers are usually very expensive.
- **Cross-Layer Design:** SDN enables an alternative approach known as cross-layer approach. In a layered structure, such as OSI reference model, this approach enhances entity integration at different layers by allowing entities to exchange information with each other. Cross-layer approaches can be developed easily on SDN platform as it offers applications an easy access to network status information.

2.5 Placement of Controller in SDN

In SDN, decoupling of the control plane from the data plane offers a more structured environment for development of new network-wide abstractions. Common SDN implementation depends on a logically centralized controller that possesses a global view of the network. However, centralized based approach for controllers has limitation in the deployment of a large-scale wireless area network (WAN) [21], which is not the subject of this study.

Furthermore, in SDN architecture, control logic of devices that process packets resides on external controllers. This introduces performance uncertainty on reliability and scalability of SDN when compared to traditional networks. Two of the most debatable questions are:

1. Can multiple controllers coexist in an SDN?
2. If yes, then where to place these controllers in the network?

Answer to the first question is yes. Multiple controllers can be used to overcome latency, *PDR* and processing delays but placement of these controllers is still a non-deterministic polynomial-time hard problem [22]. Heller et al. [23] has answered the second question by assuming different scenarios.

3. Simulations and Results

We used Mininet simulator for simulation and performance measurement. Table 1 lists the basic network simulation parameters. Multiple scenarios consisting of varying numbers of nodes and switches are considered as discussed shortly. However, the traffic generated by each node was at least 10pps and all the traffic is intended for a single sink. Furthermore, the distance between each switch varies, so that physical layout and distance can also be considered for latency. Each simulation run is of 100 seconds.

Table 1: Simulation parameters

Parameters	Value
Packet size	64KB
Switches	n (= 1, 2, 3, 4, 5)
Traffic Rate	10 - 100 pps
Simulation Time	100s each case
No of Hosts	6n

3.1 Case-1: Series of Switches

Initially, a network in Mininet is established using four switches placed in series as shown in Fig. 1, where each switch has at least 1 client associated with it. Firstly, we placed switches equidistance and later we made some slight variations in distances. We set the distance between *Switch*$_1$ and *Switch*$_3$ to 25m, and from *Switch*$_2$ to *Switch*$_0$ to 50m. Distance between *Switch*$_3$ to *Switch*$_2$ is 100m and *Switch*$_0$ to *Switch*$_1$ is 75m. This way, each packet experiences different propagation delays in each path as it traverses different switch-pair links.

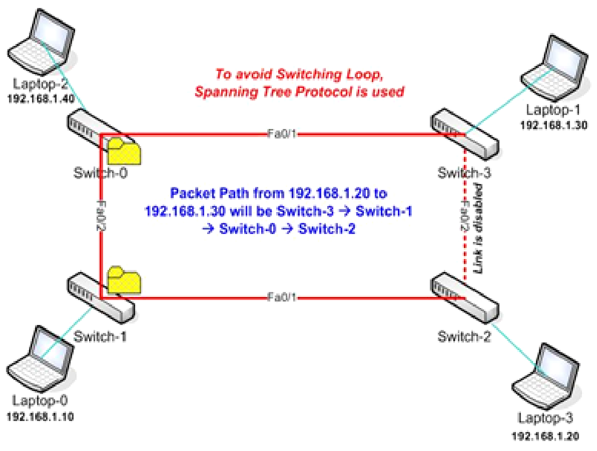
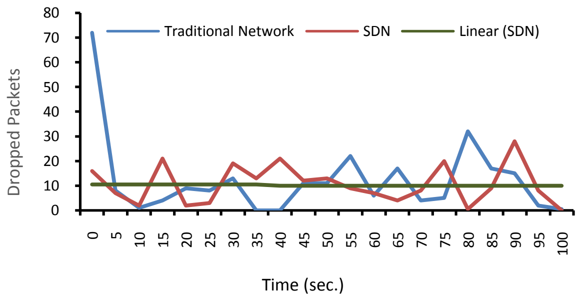

To avoid Switching Loop, Spanning Tree Protocol is used
Laptop-2 192.168.1.40
Laptop-1 192.168.1.30
Switch-0
Switch-3
Fa0/1
Fa0/2
Packet Path from 192.168.1.20 to 192.168.1.30 will be Switch-3 → Switch-1 → Switch-0 → Switch-2
Link is disabled
Switch-1
Switch-2
Laptop-0 192.168.1.10
Laptop-3 192.168.1.20

Fig. 1: A network with 4 switches using STP

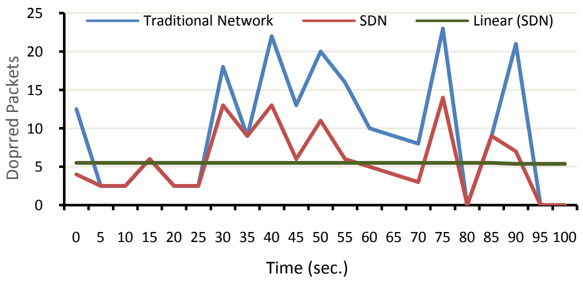
For multiple hops that a packet takes on the network to reach its destination, delay incurs. For each hop, the intermediate device/switch needs to process it so that the packet can be relayed forward accordingly, though multiple sources need to be handled simultaneously. Hence, packet processing delay and packet drops may be encountered on the network, depending upon the traffic sources, packet size and queue, etc.

The trend observed in Fig. 2 is similar, where intermediate switches of the said topology experience larger packet drops. For both types of networks, the trend is similar for Switch-0 and Switch-1, but SDN has approximately half of the packet drops as compared to traditional network. On the other hand, for Switch-3 the statistics are quite similar where SDN has some edge, but that is due to initial setup cost of traditional network. Though, SDN has shown better performance in terms of delay, latency and packet drop ratio, on close observation it is found that the latency of the varying distance among switches is approximately double as compared to traditional networks, i.e. around 10% when Fig. 2(b) and 2(c) are compared with 2(a). For Linear SDN (see Figs. 2 and 3), it has linear behavior for performance.

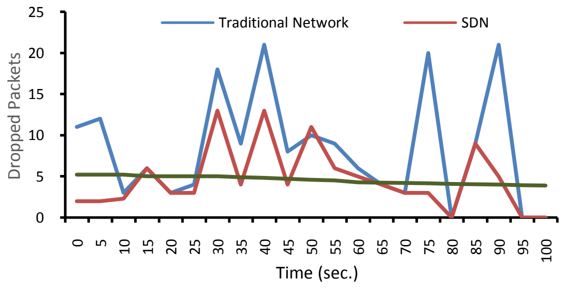
Next, for both traditional and SDN networks, initially, we started with 6 hosts and gradually increased the number of hosts and switches in order to track latency during packet transmission. Fig. 3 shows that traditional network undergoes high variations in latency at increasing hosts, especially during network initialization phase; after which it stabilizes. However, the result show periodical rise in packet drops, which is due to queuing of the packets that have to undergo multiple hops for distant destinations. For single switch scenario, where both the networks are similar, on average, at least 1 packet per second is dropped in traditional network, whereas in SDN similar behavior is observed every 5 seconds.



(a) Latency from Switch-0 to Switch-3.

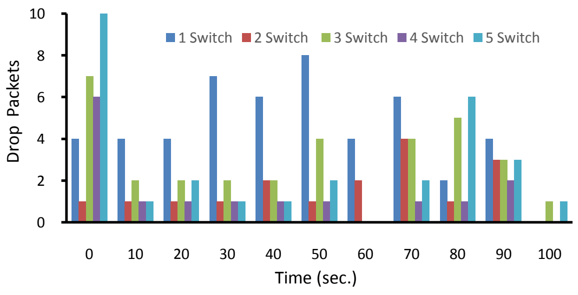


(b) Latency from Switch -1 to Switch-3.



(c) Latency from Switch -2 to Switch-3.

Fig. 2: Calculated latency variations in networks for varying distances/hops

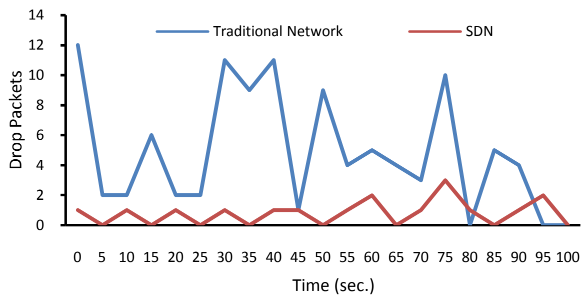


Varying Switch latency for SDN Network.

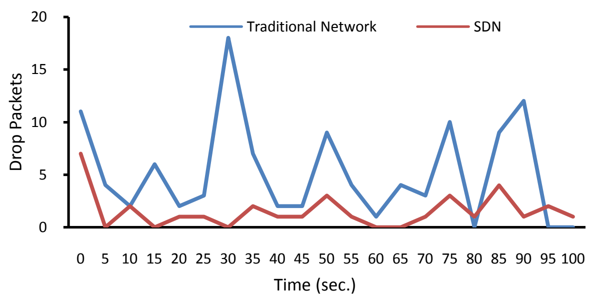
Fig. 3: Packet drops over various time windows

Though, large rise in Fig. 3 are evident in the start but thereafter none of the links have packet drops of more than 2 packets per second in SDN. When traditional network is examined separately by increasing the switch-host pairs to 2 or 3 (from one switch), the latency is doubled and quadrupled when the number of switches is increased further. For SDN, this phenomenon is quite the opposite, as latency dropped to half (on average).

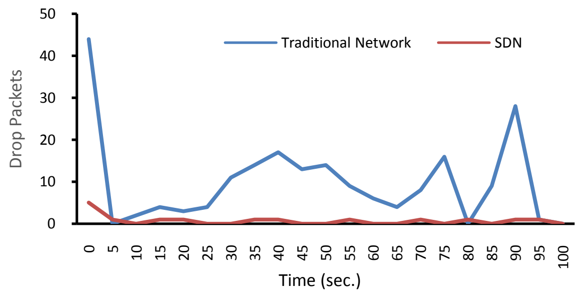
For each link, when analyzed independently (as shown in Fig. 4) the difference in latency is quite evident. For 2 or more switches where six hosts are connected on each switch; on average 80% improved performance is observed in SDN. For a 5 second interval, the best performance in terms of latency was found to be more than 90%.



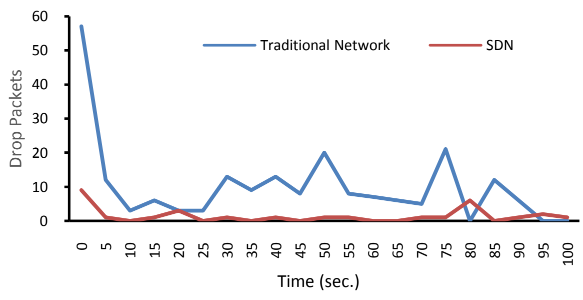
(a) 2 switches , 12 hosts.



(b) 3 switches,18 hosts.



(c) 4 switches, 24 hosts.



(d) 5 switches, 30 hosts.

Fig. 4: Latency variations for different numbers of switches and hosts

3.2 *Case-2: Multi-Layer Switch*

We have examined packet processing in another scenario as shown in Fig. 5. In this scenario, we have attached Multi-Layer Switch (MLS) to communicate with server. In this example, host Laptop-1 having IP address 192.168.1.30 communicate with server. Packet will move from Switch-0 to Switch-1 and then to Switch-2 and will be received by MLS, which forwards the packet to the server.

Fig. 6 shows the communication involved with a Multi-Layer Switch. SDN behavior is found steady, while 1 packet every 2 seconds is dropped on average, as observed in Case-1. On the other hand, steep rise in the graph for traditional network is observed during the initialization phase; afterwards the latency drops but it is still more than SDN during half of the simulation time. The traditional network stabilizes but packet drops is 2.5 times more than that of SDN (on average). SDN tends to achieve a steady state quickly, latency is distributed evenly and the pattern is quite similar throughout the simulation due to periodic overhead incurred by the control plane.

3.3 *Case-3: In-Path Router*

For multiple networks as shown in Fig. 7, involving routers for inter-network communication, the processing overhead is shown in Fig. 8.

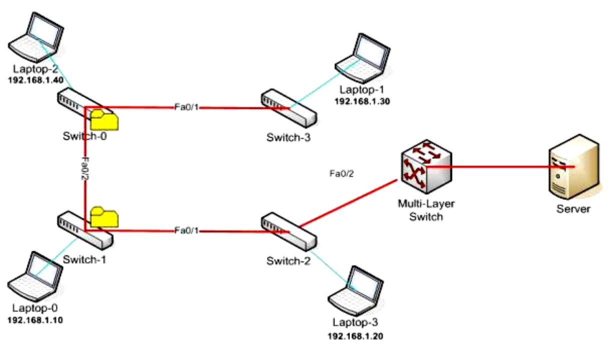


Fig. 5: Enhanced topology with a multi-layer switch

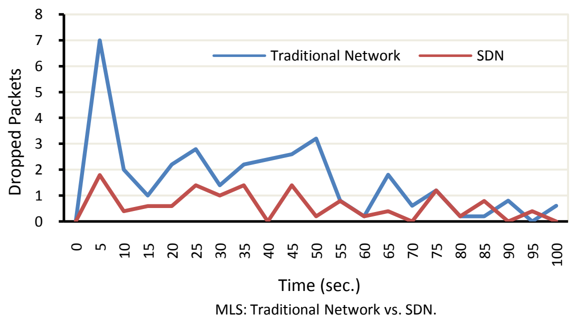


Fig. 6: Overall comparison of traditional and SDN with multi-layer switch

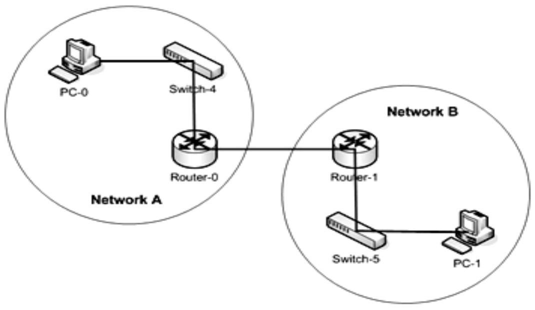


Fig. 7: Topology consisting of multiple networks connected via routers

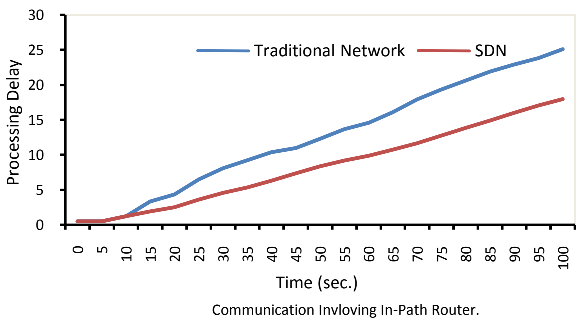


Fig. 8: Overall processing overhead for traditional networks and SDN

SDN is found to perform better as compared to traditional network in terms of processing delay. It is quite evident that except for the initialization phase, SDN outperforms throughout. The processing delays for traditional network and SDN are around 25 milliseconds and 20 milliseconds, respectively.

4. Conclusion

Traditional networks are difficult and hard to change. A key idea of SDN is the introduction of dynamic programmability feature, which overcomes this restriction. SDN makes virtualization of human mind applicable in networking by focusing on service rather than data. In this article, traditional and SDN networks are analyzed and compared from the perspective of a small/medium organization that possesses or intend to establish its own data center. The numerical results obtained clearly indicate that SDN outperforms traditional network. Possible performance penalty in functionally complex SDN infrastructure has also been studied. Results nullify such phenomenon. However, as expected, simulation experiments confirm the dependence of network performance on network topology, number of virtual nodes and available resources in hosts. Due to this, experimental results may vary in systems in terms of time and bandwidth.

In future it may be interesting to compare the behaviors of SDN networks in two platforms, Mininet and OpenDaylight, towards highlighting their limitations and strengths.

References

[1] W. Xia, Y. Wen, C.H. Foh, D. Niyato and H. Xie, “A survey on Software-Defined Networking”, IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 27-51, 2015.

[2] A. Gelberger, N. Yemini and R. Giladi, “Performance analysis of Software-Defined Networking”, Proc. of 21st Int. Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, pp. 383-393, 2013.

[3] B. Nunes, M. Mendonca, X.N. Nguyen, K. Obraczka and T. Turletti, “A survey of Software-Defined Networking: Past, present, and future of programmable networks”, IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1617-1634, 2014.

[4] C. Monsanto, J. Reich, N. Foster, J. Rexford and D. Walker, “Composing Software-Defined Networks”, Proc. of 10th USENIX Conference on Networked Systems Design and Implementation, pp.1–14, 2013.

[5] D. Levin, A. Wundsam, B. Heller, N. Handigol and A. Feldmann, “Logically centralized? State distribution trade-offs in Software Defined Networks”, Proc. of 1st Workshop on Hot Topics in Software Defined Networks, pp. 1-6, 2012.

[6] M.F. Bari, A.R. Roy, S.R. Chowdhury, Q. Zhang, M.F. Zhani, R. Ahmed and R. Boutaba, “Dynamic controller provisioning in Software Defined Networks”, Proc. of 9th Int. Conference on Network and Service Management, pp. 18-25, 2013.

[7] M. Casado, M.J. Freedman, J. Pettit, J. Luo, N. McKeown and S. Shenker, “Ethane: Taking control of the enterprise”, ACM SIGCOMM Computer Communication Review, vol. 37, no. 4, pp. 1-12, 2007.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker and J. Turner, “OpenFlow: Enabling innovation in campus networks”, ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008.

[9] OpenFlow Switch Specification, User’s and Theory Reference Manual, 2009.

[10] OME Committee, “Software-Defined Networking: The new norm for networks”, Open Networking Foundation, 2012.

[11] J. Medved, R. Varga, A. Tkacik and K. Gray, “OpenDaylight: Towards a model-driven SDN controller architecture”, Proc. of 15th Int. Symposium on A World of Wireless, Mobile and Multimedia Networks, pp. 1-6, 2014.

[12] S.Y. Wang, C.L. Chou and C.M. Yang, “EstiNetOpenFlow network simulator and emulator”, IEEE Communications Magazine, vol. 51, no. 9, pp. 110-117, 2013.

[13] Banjar, P. Pupatwibul, R. Braun and B. Moulton, “Analysing the performance of the OpenFlow standard for software-defined networking using the OMNeT++ network simulator”, Proc. of Asia-Pacific Conference on Computer Aided System Engineering, pp. 31-37, 2014.

[14] K. Sood, S. Yu and Y. Xiang, “Performance analysis of Software-Defined Network switch using M/Geo/1 model”, IEEE Communications Letters, vol. 20, no. 12, pp. 2522-2525, 2016.

[15] H. Farhday, H.Y. Lee and A. Nakao, “Software-Defined Networking: A survey”, Computer Networks, vol. 81, no. 2, pp. 79-95, 2015.

[16] F. Hu, Q. Hao and K. Bao, “A survey on Software Defined Networking (SDN) and OpenFlow: From concept to implementation”, IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2181-2206, 2015.

[17] M.I. Lali, M.M. Bilal, M.S. Nawaz, B. Shahzad and S. Khalique, “Effect of input-output (IO) buffering to minimize flow control blocking in Software Defined Networking”, The Nucleus, vol. 53, no. 3, pp. 208-213, 2016.

[18] B. Lantz, B. Heller and N. McKeown, “A network in a laptop: Rapid prototyping for Software-Defined Networks”, Proc. of 9th ACM SIGCOMM Workshop on Hot Topics in Networks, pp.1-6, 2010.

[19] D. Patterson, “Latency lags bandwidth”, Communications of the ACM, vol. 47, no. 10, pp. 71-75, 2004.

[20] B. Forouzan, Data Communications and Networking, 5th Edn., McGraw-Hill, NY, USA, 2012.

[21] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado and R. Sherwood, “On controller performance in Software-Defined Networks”, Proc. of 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, Enterprise Networks and Services, pp.10-10, 2012.

[22] M. Guo and P. Bhattacharya, “Controller placement for improving resilience of Software Defined Networks”, Proc.of 4thInt. Conf. on Networking and Distributed Computing, pp. 23-27, 2013.

[23] B. Heller, R. Sherwood and N. McKeown, “The controller placement problem”, Proc. of 1st Workshop on Hot Topics in Software Defined Networks, pp. 7-12, 2012.