The Nucleus **53**, No. 3 (2016) 208-213

www.thenucleuspak.org.pk

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

Effect of Input-Output (IO) Buffering to Minimize Flow Control Blocking in Software Defined Networking

M.I. Lali[1], M.M. Bilal[1], M.S. Nawaz[2*], M. Deen[1], B. Shahzad[3] and S. Khaliq[1]

[1]*Department of Computer Science & IT, University of Sargodha, Sargodha, Pakistan*

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

[3]*College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia*

drikramullah@uos.edu.pk; mussadiqbilal@yahoo.com; msaqibnawaz@pku.edu.cn; moinudeen@gmail.com; bshahzad@ksu.edu.sa; askhalique@gmail.com

ARTICLE INFO

Article history :
Received : 21 July, 2016
Revised : 02 September, 2016
Accepted : 20 September, 2016

Keywords :
Software defined networking
Input output buffers
Latency
Packet delivery ratio

ABSTRACT

Software Defined Networking (SDN) is a new network technology that tries to overcome the shortcomings and problems in traditional networks. By centralizing the network state in control layer, SDN architecture makes it easier to define and enforce consistent policies across the network. By separating data plane from control plane, dynamic requirements of complex networks can be easily manage through SDN. For future networks, SDN is already playing a significant role. SDN offers robustness, flexibility, vendor's independent platform and consistent policies across the network. In this article, we analyze different Input-Output (IO) buffering strategies which effect flow control blocking in SDN. We evaluate the performance of three different IO buffering strategies: single buffering, double buffering and ring buffering. Analyses of three IO buffering strategies are carried out in terms of two network parameters: latency and packet delivery ratio (PDR). Our results show that in terms of latency, ring buffering performs much better than single and double buffering for varying number of hosts and switches in a network. Furthermore, PDR is significantly minimized by use of buffers during network congestion in SDN.

1. Introduction

Emerging trends in mobile, social networks, cloud computing and big data have given new challenges to future Internet, for which ubiquitous accessibility, high bandwidth and dynamic management are crucial [1]. In conventional computer networks, the data plane and control plane operate together over the same channel for transmission process. In such networks, control information and user data is sent over the same channel and it is difficult to maintain data and information on same channel. Furthermore, various problems arise such as bad signaling, reset phone trunk lines, inefficient network control, network maintenance and up-gradation [2]. Conventional networks do not meet the current requirements of end users and faces a-lot of challenges in configuration of network devices, adding new functionality and controlling the network behavior from a single point. In order to mitigate flexibility problems, researchers have invested an initiative that implements networks with greater programming capabilities and reduce the need to replace switching equipment [3]. These requirements lead to the development of new paradigm in networking known as Software Defined Networking (SDN).

SDN [4] is an evolving network structural design which allows network applications to be controlled and maintained from a single software module called SDN controller [5]. SDN is changing the technique of how we plan, maintain and manage networks and is one of the most considerable network architecture that shifts the networking industry in past few years. SDN has a potential to meet the new network challenges from rising trends of cloud computing, Internet of things and big data applications. The goal of SDN is to introduce network innovations, automation and managing larger networks from a single entity. By separating the data plane from control plane and moving the control plane to a centralized controller, SDN offers network operators a strong capability to deploy a wide-range of network policies (such as routing, fault-tolerance and security) along with the ability to implement new network technologies [6]. Additionally, the network administration is much more focused in terms of applications and services rather than topologies and data management. More recently, the development of Ethane [7] and OpenFlow [8] have brought the implementation of SDN closer to reality.

[*] Corresponding author

OpenFlow is an open standard protocol that provides researchers to deploy and run new protocols in SDN. OpenFlow is the first standard interface and a secure channel through which we can interact with different layers in SDN [8]. Network devices such as routers and switches could be accessed through OpenFlow. Several recent papers have studied particular compositional parts of SDN [9]. OpenFlow allows us to perform operations such as inspecting and modifying flow table entries on network devices. The OpenFlow protocol is used as a key and standard protocol in SDN to interact with the forwarding plane of network devices. The Openflow specification [11] builds up the rules of communication between a controller and data plane. The Open Networking Foundation (ONF) [12] brings together about 90 companies and is dedicated to releasing, promoting and adoption of OpenFlow specification. Standardization may advance more through new open source associations OpenDaylight [13] and Estinet [14] projects.

Though, numerous surveys and theoretical literature exists to date [1, 10, 15, 16] which highlights various concepts of SDN. Furthermore, in recent years, the performance of SDN with different parameters and scenarios is evaluated in [3, 17-20]. However, still it is not clear which are the different Input Output (IO) buffering strategies that can identify blocking flows, which help in minimizing the flow control blocking in SDN. Flow control blocking and congestion control minimization is essential in both traditional and SDN networks in order to achieve high throughput and minimum packet loss. In SDN, controller is used to manage network flow control. SDN controllers used protocols such as OpenFlow for directing switches to where send packets. Buffer is a place which is shared by networking devices that are working at different speed. Buffer permits every networking device to work without being stalled by some other device in the network. Buffering strategies are used for many purposes, including: holding data temporarily in the network, sending/receiving data over the network, adaptive and video streaming over the Internet and reduce the impact of data rate variability during message rate spikes. Numerous factors play a vital role for effectiveness of buffer such as buffer's size and selection of algorithm for transmission of data. In this study, we looked at different buffering strategies and their effect on blocking in SDN. We have evaluated the performance of SDN by using three different IO buffering strategies (single buffering, double buffering and ring buffering), compare the results and identify a strategy that performs better in minimizing blocking flows in SDN. Performance of SDN under mentioned buffering strategies is evaluated in terms of latency and packet delivery ratio (PDR).

Mininet [17] is used for simulation of IO buffering strategies in SDN along with different topologies. Mininet is an innovative process based network emulator for prototyping large networks, evaluating performance and bandwidth usage on a single computer or laptop. In Mininet, hosts, links, switch and SDN controllers look like a complete network and users can implement and test new network features or new architecture. Furthermore, same code and test scripts can be deployed in a real production network. We use Mininet python API programming language for making custom topologies and scripts.

The rest of the article is organized as follows: Three IO buffering strategies and network parameters that we used for performance evaluation are discussed in Section 2. Simulation experiments are performed in Section 3, where SDN performance is checked and discussed for different IO buffering strategies. Finally, we conclude the article and highlight future directions in Section 4.

2. IO Buffering Strategies and Network Performance Measures

In SDN, network can be managed and controlled quickly by using a centralized controller. The use of IO engine in SDN controller offers fast response time and higher flow setup rate than current SDN controllers. IO engine uses multi-queue design approach for getting high performance, which differentiates it from other SDN controller's approach. IO engine is the best among current SDN controllers in processing core scalability issues [21]. Congestion in networks generally occurs when a host is receiving more data (packets) than it can handle [22]. Congestion can reduce overall quality of service and other effects of congestion include loss of packet during transmission, queuing delay and blocking of new connections. Main aim of congestion control is to keep the number of packets below a defined level at which network overall performance dramatically decreases [23]. One of the congestion control mechanism is to use buffers during network congestion. A buffer is used to temporarily store data during transmission. IO buffering strategies that are considered in this work are:

- **Single Buffering:** In single buffering, we have used only one buffer to hold the packets during bottleneck so that latency and delay rates of transmitted packets could be minimized. The size of the buffer is equal to $RTT \times C$ where RTT is the round trip time of the packets transmitted over the connection and C is the data rate of the bottleneck link.
- **Double Buffering:** In double buffering, we have used two system buffers instead of one. In this strategy, we double the size of buffer so that it can absorb more packets and delay rates of transmitted packets minimized.
- **Ring Buffering:** In this technique, more than two buffers have been used. Each individual buffer is one

unit in a circular buffer. The block diagram for ring buffering is shown in Fig. 1, where incoming data/packet is stored before the output. Generally, data is stored in the buffer when it is retrieved from an input device or when data is sent to an output device.Ring buffering works on the basis of first-in-first-out (FIFO) method, where the oldest (first) packet in the buffer is processed first.

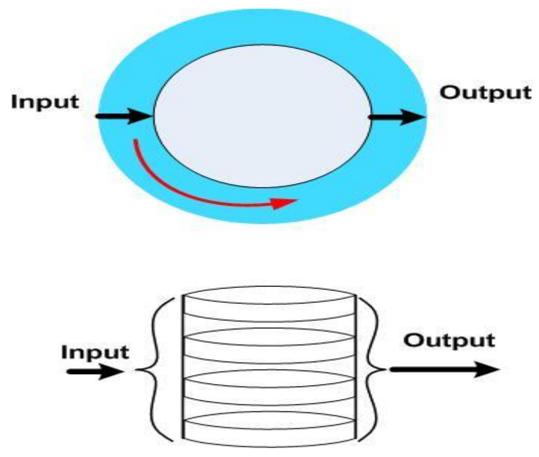

Fig. 1: Working of ring buffering

We have used latency and PDR as two network parameters for performance comparison of above mentioned strategies in SDN.

- **Latency:** Bandwidth and latency are the two key elements that contribute to network speed. Latency has enormous effect on the system speed. Systems associated with low latency are the one which encounters little delay, while a high inactivity association experiences long postpones. Actual network bandwidth generally varies over time and is affected by high latencies. Excessive latency introduces bottlenecks that will stop data from filling the network pipe, thus bandwidth is decreased. Impact of latency on network bandwidth depends on the delays source and this impact can be temporary (that lasts for few seconds) or it can be persistent [24].
- **Packet Delivery Ratio (PDR):**The ratio of delivery of packets from the point of start to the destination is called PDR. In other words, to what extent data is delivered to the destination. This ratio can also be determined by the ratio of successfully delivered packets to destination vs. the number of packets which were sent [25].

3. Simulations and Results

We used Mininet for simulation and performance measurement. Table 1enlists the basic network simulation parameters. Multiple scenarios consisting of varying numbers of node and switch are considered that are discussed shortly. However, the traffic generated by each node was at least 10pps. 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, link bandwidth is 10 MBps and buffer size is $RTT \times C$.

Table 1: Simulation parameters

Parameters	Value
Packet size	64KB
Switches	$F_n = 2^n$
Traffic Rate	10-100 pps
Simulation Time	100s each case
Link Bandwidth	10 MBps
Buffer Size	$RTT \times C$
No of Hosts	$F_n = 6(n)$

Initially, a network in Mininet was established using 1 switch where six clients are attached with it: *H1, H2, H3, H3, H4, H5* and *H6*. We gradually increased the number of switches and hosts in order to track out the latency and delay rates during the packet transmission with three IO buffering strategy. We have experienced different latency stats for packets with the interval of 5 seconds. We used IO buffers at control plane to identify blocking flows in order to minimize flow control blocking. Fig. 2 shows the latency and delay rate difference in a network having one switch and six hosts. In ring buffering technique, latency time for packets are minimized significantly and having less certain rises as compared to single and double buffering approaches.

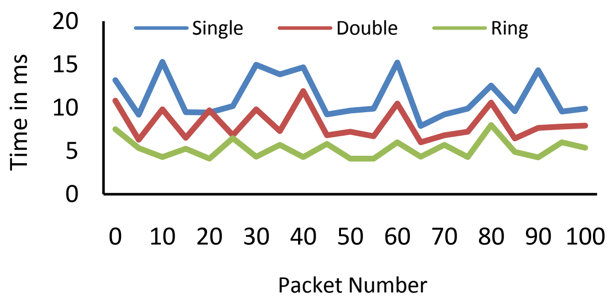

Fig. 2: Latency variations of six hosts and one switch network

We progressively expanded the number of hosts and switches to track out latency during packet transmission. Fig. 3 shows the latency for 2, 4 and 8 switches respectively. For network with more than one switches, the results shows certain rise which may be an outcome in queuing of the packets that are of longer distance across the networks. Similarly, latency time of delivered packets significantly improves in ring buffering technique with increase in the total number of switches and hosts.

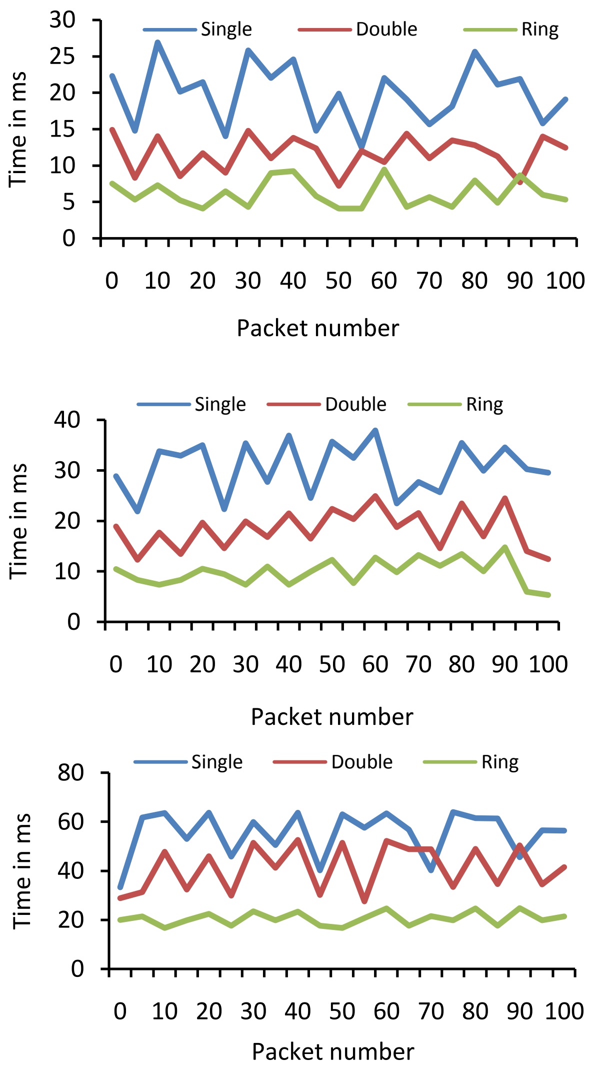

Fig. 3: Calculated latency variations in networks for (a) 2 hosts and 2 switches network, (b) 16 hosts and 4 switches network and (c) 16 hosts and 8 switches network

For 4 switches, ring buffering technique performance improved 30% with respect to the other techniques, whereas, ring buffering technique shows 50% improved performance for 8 switches network. We can say that performance of ring buffering technique improves for complex network structure. Furthermore, evaluation of three buffering techniques with 1 switch performs quiet similarly. However, periodically the result shows certain rise which may be a result in queuing of the packets in single and double buffering approaches.

We also compared three buffering techniques for latency in each strategy as shown in Table 2 and Fig. 4. Average latency time in single, double and ring buffering strategies for varying number of switches is listed in Table 2. As the network become more complicated and dense, average latency time in each buffering strategies increases.

Table 2: Comprehensive comparison of three buffering techniques

Network Topology	Avg. Latency in Single Buffering	Avg. Latency in Double Buffering	Avg. Latency in Ring Buffering
1 Switch	11.86 ms	8.52 ms	5.42 ms
2 Switch	20.87 ms	12.23 ms	6.45 ms
4 Switch	32.08 ms	19.24 ms	10.33 ms
8 Switch	55.09 ms	42.17 ms	21.55 ms

A packet travels multiple hops in the network to reach its destination. At each hop, intermediate device or switch process the packet in order to move the packet forward. Hence, packet processing delay and drop of packets may be encountered on the network depending upon the traffic sources, packet size and queue, etc. The effect of multiple switches for each buffering strategy is shown in Fig. 4. Furthermore, Fig. 4 illustrates that latency increases with increase in number of switches in SDN network. In single buffering (Fig. 4a), latency for 1 and 2 switches is low at packet 70. Latency in single buffer for packet number 70 and 90 is also low for 8 switches. In addition, there is not much variation in latency for single buffering in multiple switches. Latency variation for double bufferingand ring buffering is shown in Fig. 4b and Fig. 4c respectively. In Fig. 4b, low latency is encountered for 1 switch at packet number 50, 70, 90 and 100. At packet 50 and 90, low latency is encountered for 2 switches and low latency is encountered for 4 switches at the end of the packet transmission in the network. Similarly, in ring buffering, low latency is encountered for 2 switches at the start of transmission and at packet 50. For 8 switches, low latency is encountered at network initialization time and in the middle of transmission.

Generally, network traffic is inherently bursty and as a result, all network switches have buffer queues that absorb additional packets that arrive during traffic bursts. The buffers then drain during traffic congestion, keeping average utilization of outgoing links high and causing lower packet drops. We used buffer so that packet drop ratio could be minimized in case of congestion. First, we calculated PDR without using buffer and send 50 packets in total during a simulation run of 100 seconds. Out of 50 packets, 23 packets were dropped and PDR is calculated with following formula:

$$PDR = \frac{\#\text{ of packet received}}{\#\text{ of packet sent}} \tag{1}$$

PDR = 27/50 = 0.54 in case of using no buffer.

Now with the same scenario, we attached a single buffer and checked PDR. In single IO buffering strategy, 11 packets were dropped. PDR for single buffering strategy is also calculated from formula 1. PDR = 39/50 = 0.78. In order to increase the PDR, we have used double buffering strategy. So that maximum number of packets absorbed by the buffers and packer dropped ratio could be minimized. In total, 8 out of 50 packets were dropped with double buffering strategy. For double buffering strategy, PDR =42/50 = 0.84. For ring buffering strategy, 2 packets were dropped and calculated PDR = 48/50 = 0.96.

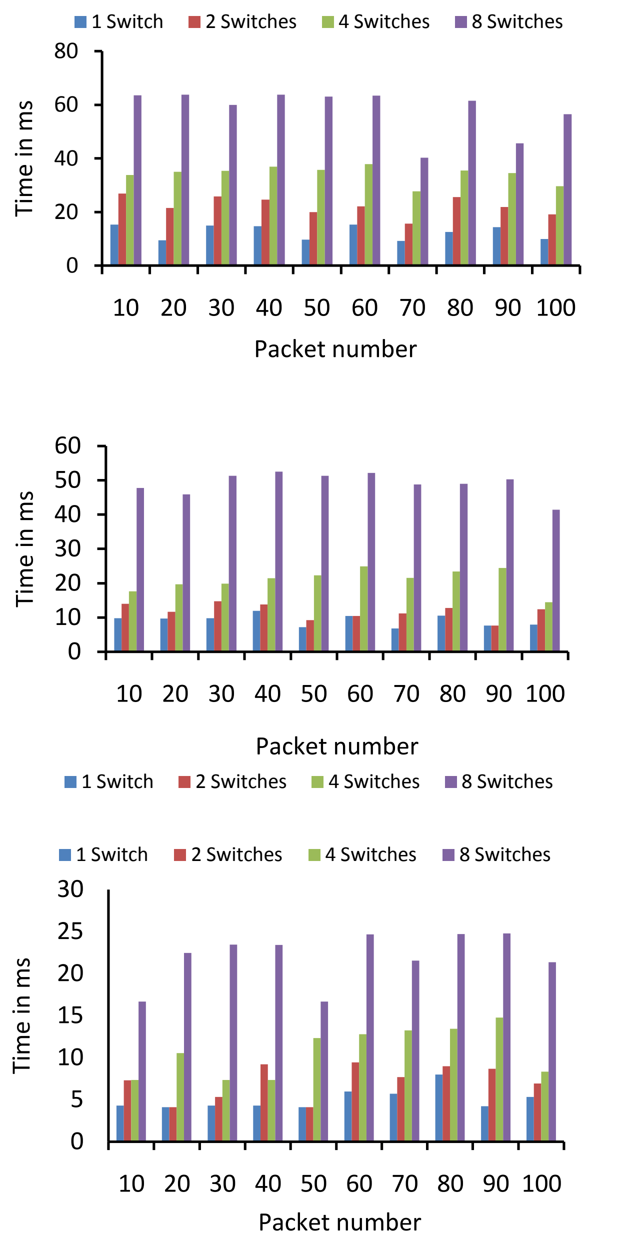

Fig. 4: Varying switch latency for (a) single buffering, (b) double buffering, and (c) ring buffering

In terms of PDR, ring IO buffering strategy shows almost 20% better performance than single IO buffering strategy and 12% better performance than double IO buffering strategy.

4. Conclusion

Legacy networks are troublesome and difficult to control. On the other hand, SDN truly made networks simpler and easy to control. SDN is now most utilized methodology as a part of systems administration. In this article, we studied the performance of different IO buffering techniques to minimize blocking in SDN during heavy network traffic. We analyzed three buffering techniques i.e. single buffering, double buffering and ring buffering at control plane. We used different test cases in Mininet for performance comparison of IO buffering strategies. Our results showed that ring buffering technique outperformed both single and double buffering strategies. We also found that the use of buffers is very helpful during congestion in SDN and packet drop ratio in case of congestion could be significantly minimized by the use of buffers in SDN. Calculated PDR for single and double buffering strategy is 78% and 84% respectively. In ring buffering technique, we got much better results and approximately 96% packets successfully delivered to their destination hosts.

One interesting area for future is to compare the performance of IO buffering strategies in other SDN emulators such as EstiNet and OpenDaylight. Other interesting area would be the development of high-performance IO engine that offers fast response time and higher flow setup rate than state-of-the-art SDN controllers.

References

[1] W. Xia, Y. Wen, C.H. Foh, D. Niyato and HaiyongXie, "A survey on software-defined networking", IEEE Communications Surveys & Tutorials", vol. 17, no. 1, pp. 27-51, 2015.

[2] F. Zhao, D. Zhao, X. Hu, W. Peng, B. Wang and Z. Lu, "A 3N Approach to network control and management", Proc. of 26th Int. Parallel and Distributed Processing Symposium Workshops & PhD Forum, pp. 1237-1242, 2012.

[3] A. Gelberger, N. Yemini and R. Giladi, "Performance analysis of software-defined networking (SDN)", Proc. of 21th Int. Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems, pp. 389-393, 2013.

[4] H. Kim and N. Feamster, "Improving network management with software defined networking", IEEE Communications Magazine, vol. 51, no. 2, pp. 114–119, 2013.

[5] S. B. Brezetz, G. B. Kamga and M. Tazi, "Trust support for SDN controllers and virtualized network applications", Proc. of 1st IEEE Conf. on Network Softwarization, pp. 1-5, 2015.

[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. Conf. 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.mTurner, "OpenFlow: Enabling innovation in campus networks", ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74, 2008.

[9] P. Lin, J. Bi, Z. Chen, Y. Wang, H. Hu, and A. Xu, "WE-bridge: West-East Bridge for SDN inter-domain network peering", Proceedings of Computer Communications Work-shops (INFOCOM WKSHPS), pp. 111-112, 2014.

[10] 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.

[11] OpenFlow Switch Specification, Version 1.1.0, pp. 1–56, 2011.

[12] Software-Defined Networking: The New Norm for Networks, Open Networking Foundation, White Paper. [Online], https://www.openn etworking.org/

[13] 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.

[14] 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.

[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] B. Lantz, B. Heller and McKeown, "A network in a laptop: Rapid prototyping for software defined networks", Proc. of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, pp. 1-6, 2010.

[18] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll and P.Tran-Gia. "Modeling and performance evaluation of an openflow architecture", Proc. of 23rd Int. Teletraffic Congress, pp. 1-7, 2011.

[19] C.N. Shivayogimath and N.V. Uma Reddy, "Performance analysis of a software defined network using mininet", Artificial Intelligence and Evolutionary Computation in Engineering Systems, pp. 391-398. 2016.

[20] 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, and Enterprise Networks and Services, pp.10–10, 2012.

[21] S. H. Park, B. Lee, J. Shin and S. Yang, "A high-performance IO engine for SDN controllers", Proc. of 2014 Third European Workshop on Software Defined Networks, pp. 121-122, 2014.

[22] P. Gevros, J. Crowcroft, P. Kristein and S. Bhatti, "Congestion control mechanism and the best effort service model", IEEE Networks, vol. 15, no. 3, pp. 16-26, 2001.

[23] W. Stallings, "Data and Computer Communications", Prentice-Hall Publishers, 2007.

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

[25] B. Forouzan, "Data Communications and Networking", Fifth Edition. McGraw-Hill, NY, USA, pp. 85-98, 2012.