Analysis and Performance Evaluation of SDN Queue Model

Samuel Muhizi[1], Gregory Shamshin[1], Ammar Muthanna[1], Ruslan Kirichek[1,2](✉), Andrei Vladyko[1], and Andrey Koucheryavy[1]

[1] The Bonch-Bruevich State University of Telecommunication, 22 Prospekt Bolshevikov, St. Petersburg, Russia

[2] Peoples' Friendship University of Russia (RUDN University), 6 Miklukho-Maklaya Street, Moscow, Russia

samno1@yandex.ru, reignsword@gmail.com, ammarexpress@gmail.com, {kirichek,vladyko}@sut.ru, akouch@mail.ru

Abstract. In this paper, we present an Openflow-SDN based network visualization and performance evaluation model that helps in network designing and planning to examine how networks' performance will be affected as the traffic loads and network utilization change. To achieve the aimed goal, as a research method, we used AnyLogic Multimethod simulation tool. This is a first of its kind where SDN performance evaluation is based on queuing model simulation to monitor change of average packet processing time for various network parameters. Using presented in this work SDN model, network administrators and planners can better predict likely performance changes arising from traffic variation. This allows them to make prompt decisions to prevent seemingly small issues from becoming major bottlenecks.

Keywords: SDN controller · OpenFlow switch · Flow table · AnyLogic · Queue model · Simulation model · Analytical model

1 Introduction

The concept of software defined networks (SDN) is a growing trend in the domain of telecommunication network management, that offers to remove restrictions on existing infrastructure networks by dividing the network control plane and data plane by means of transferring control functions of network forwarding devices (routers, switches) in the applications running on a single entity (controller), which makes the network more centralized and improves its software management capabilities [1–3]. This simplifies the network usability and significantly reduces the network system cost and equipment. However, all these benefits come with a non-negligible problem in network functionality such as packet transmission rate and network performance, which are attributed to the use of the controller as a remote system to manage all transmission network devices (switches). The controller can manage one or several OpenFlow switches; it contains a network operating system that provides network services for low-level network management, network segments, and the state of network elements and applications, implementing a high-level network management and data flow. Each controller has at least one application that manipulates switches connected to it, and can provide a global view of physical network topology under the controller management. The idea of creating a unified, independent from the network equipment manufacturer [4–6], program-controlled interface between the controller and the network transport infrastructure is defined in the OpenFlow standards and OpenFlow protocol [7], which allows users to define and control with whom, under what conditions and with what quality can interact on the network. OpenFlow is an open standard, which describes the remote management requirements to a switch that supports OpenFlow protocol. According to OpenFlow standard specifications, the interaction between the controller and the switch is carried out through OpenFlow protocol. Each switch must contain one or more flow tables and group tables, which perform packet lookups and forwarding, and support OpenFlow secured channel to a remote controller. Each flow table in the switch contains a set of flow entries; each flow entry consists of match fields, counters, and a set of instructions to apply to matching packets. Data management in OpenFlow is carried out, not on the individual packet level, but at the level of their packet streams. Rules are dynamic. Packets which have no match are sent to the controller (packet in). Controller creates appropriate rule and sends packet back to switch (packet out) for processing. The rule in the switch is set only for the first packet, and then all the other packets of the flow use it.

In this paper is introduced an analytical and simulation analysis of an SDN model where the network is modelled as queueing system to capture the time costs associated with the controller and the switches activities.

2 SDN Modelling Concept

In case of SDN deployment, the controller usually manages multiple OpenFlow switches each connecting group of host. The typical SDN architecture is shown in Fig. 1. The OpenFlow switch packet forwarding procedure is shown in Fig. 2. The switch performs flow table lookups on packet arrivals. If the lookups succeed, the switch applies the actions in the matched table entry to the packet, typically forwarding it to the specified interface. Otherwise, the packet is supposed to belong to a new flow, and the switch sends it to the upper SDN controller in a packet-in message. The controller defines the corresponding flow rules and sends to the switch it in packet-out or flow_mod packet. As a result, the SDN controller receives a flow of packet-in messages from each OpenFlow switch.

In the modern telecommunication networks, at the same time are processed different types of information (video and audio information, compressed video and audio information, and less sensitive to delay data) with different quality of service levels. Accordingly, traffic management methods play a key role in its optimization and to minimize network losses. To create a network model (topology and its elements characteristics), and dynamically simulate its operation mechanism, optimize its characteristics, analyze and manage the traffic, evaluate the performance bounds undoubtedly, it is more than necessary to use more advanced multimethod modeling and simulation tools for complex systems research [8–10].

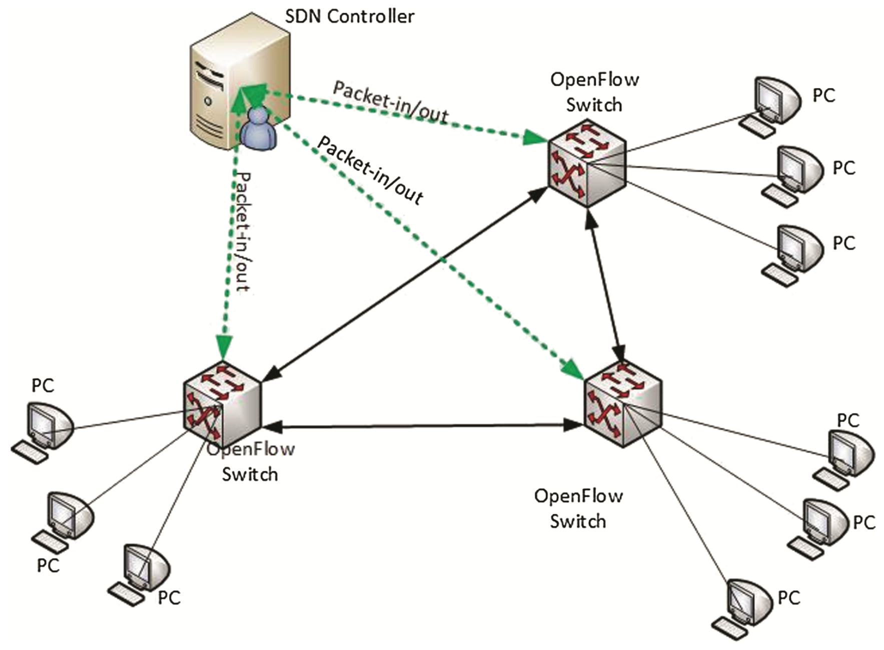

Fig. 1. A typical SDN architecture.

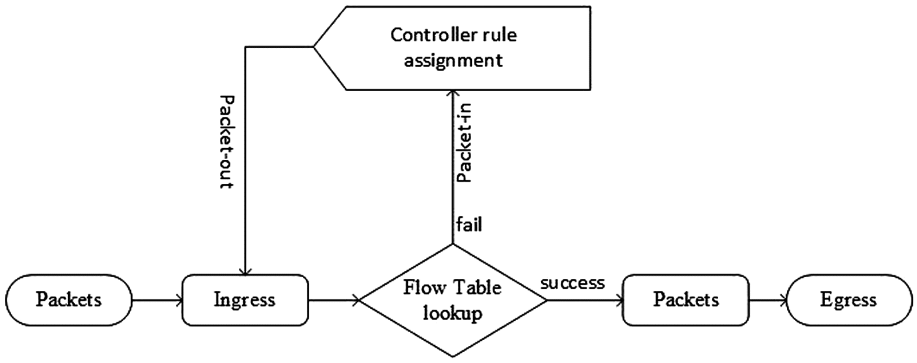

Fig. 2. The packet forwarding of an OpenFlow switch.

3 Model Description

Operations on SDN switches with different data transmission processes control capabilities lead to a significant delay change for flow entries configuration. In order to simulate different general arrival processes, describe and monitor flow exchange between the controller and the switches we presented the SDN as queuing system. The SDN queue model was built using AnyLogic Multimethod simulation modeling tool and is shown in Fig. 3. AnyLogic is a very flexible [11], dynamic simulation tool with high flexibility and unlimited expansion possibilities due to operating system-independant native Java environment. AnyLogic runs on Windows, Mac and Linux. AnyLogic initially designed to support multiple modeling methods and their arbitrary combinations it offers the modeler more flexibility than any other existing simulation tools by the means of reduction of development costs and -times: fast integration of pre-configured simulation elements with the comprehensive object libraries. It makes it possible to design various types of models with one single tool: agent-based, system dynamic, event-oriented, continuous or dynamic models.

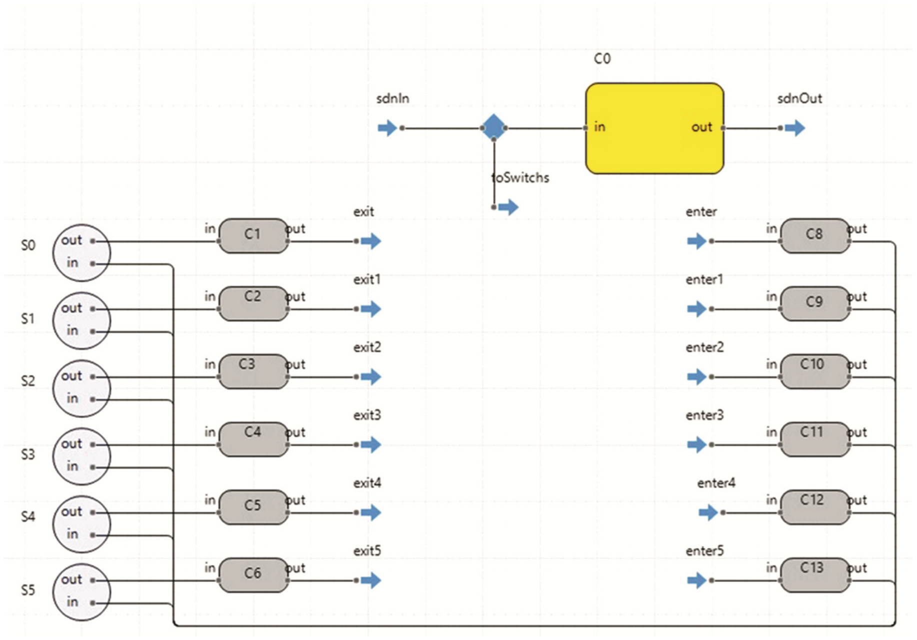

Fig. 3. SDN conceptual model in AnyLogic 7.3.6

The SDN model consists of 6 OpenFlow switches and a controller. Switches are numbered from S0 to S5 and the controller C0. Source objects generate incoming traffic with a specified arrival rate at each and every switch connected to controller.

3.1 Packet Processing on the Switch

Figure 4 shows a switch packet forwarding process. For a more accurate simulation of the information exchanges between the switch and the controller, incoming packets were divided in four categories; each of them is determined by specified occurrence probability. From the switch, packets to the controller are sent using specific communication channel. In this model the channel processing unit is built that it doesn't significantly affect on the overall evaluation of the model performance.

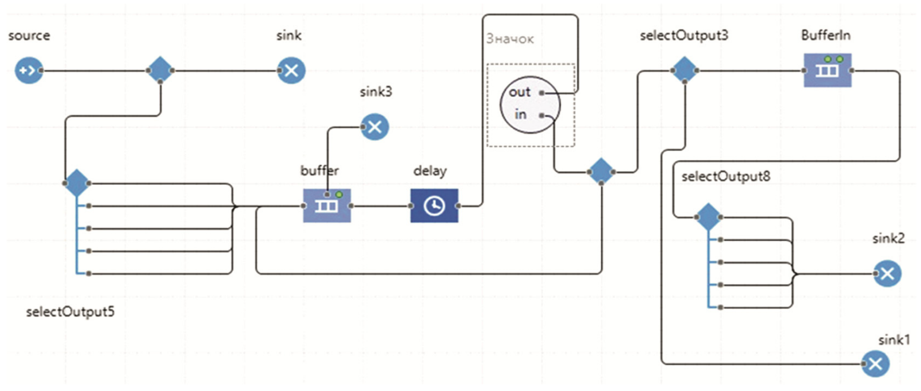

Fig. 4. OpenFlow switch conceptual model in AnyLogic 7.3.6

3.2 Packet Processing on the Controller

The controller manipulates multiple switches and therefore the arrival process of packets at the switch characterizes the packet processing rate at the controller. Accordingly, the model describes the packet transmission process management at different SDN model service units. We consider SDN controller model as shown in Fig. 5.

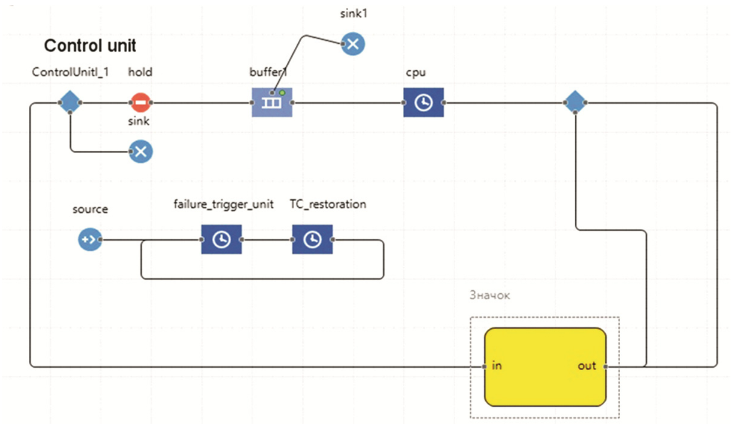

Fig. 5. SDN controller conceptual model in AnyLogic 7.3.6

Packet-in and Packet-out messages. On the arrival of a packet, the switch checks the flow table for presence of the destination address. The SDN model contains a central unit (controller) manage flow entries in the network. In this model switch numbers match the source and destination addresses. Forwarding rules and destination addresses for arrival packets are defined for each packets category arriving at the switch. First packet of the category is sent to the controller for rules and destination address confirmation. The controller determines the rule to assign to the respective packet category and sends it back as a response to the packet-in message to the switch. The switch then forwards the packet to the destination address. The next packets in this category then use the assigned forwarding rules and there won't be any requirement to confirm addresses. The designed controller model works with a given failure rate. If the failure occurs, the entire flow entry checking process starts again.

4 Modeling Results

Using the above described SDN model, we measured the network load, therefore for given parameters a network administrator can simply establish required quality of service for different network nodes by managing the delay, delay variation (jitter), bandwidth, and packet loss parameters on a network. Experimental processes were performed using a modeling time of 3600 s, and the memory size for the simulation was set to 1024 MB.

To illustrate the impact of various network parameters on the quality of service exist multiple options: arrival traffic rate changes, trigger sequences of packet-in messages, controller performance impact on the overall packet processing mean time, etc. In Fig. 6, the plot highlights the switch average packet processing time for different packet arrival rates. The more packet arrival rate increases, the more average packet processing time increases, thus the increasing arrival traffic rate will result in network throughput decrease. The plot can be used to determine the maximum load that the network should reach before its performance is compromised. For a fixed packet arrival rate on each switch we measured changes in average packet processing time while increasing the controller service rate. The simulation results are shown in Fig. 7. The average packet processing time significantly decreases as controller service rate increases. Therefore, the network throughput increases.

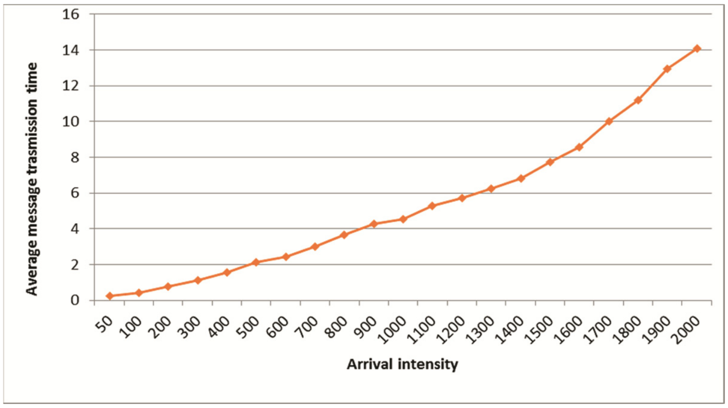

Fig. 6. Average packet processing time of switches

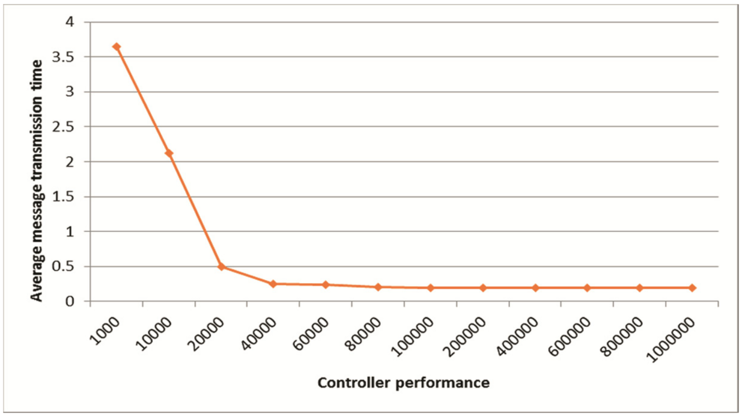

Fig. 7. Average packet processing time of the controller

5 Analytical Modeling Framework

To assert the described above SDN model we proceeded to analytically evaluation of OpenFlow switch. For that we considered a queueing model [12] for OpenFlow-based SDN [13–15] as illustrated in Fig. 8. The switches and controller are modelled as queueing systems to capture the time cost of the network.

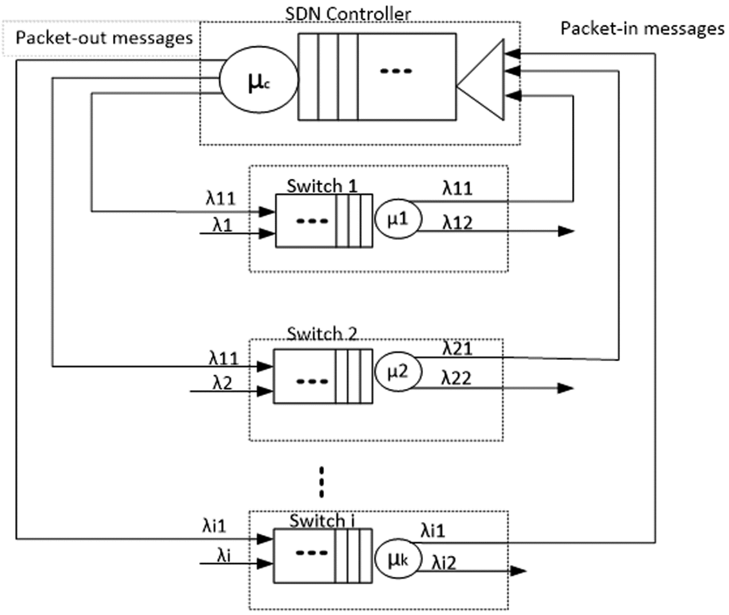

Fig. 8. OpenFlow-based SDN queueing model

We assume that the packet arrival process in the network follows a Poisson Process and the average arrival rate in the ith switch is λi, and that the arrivals in different switches are independent. Packets may not match any flow entries in which case they are forwarded to the controller via packet-in message. This happens with probability ρ. Packets are classified into two classes, both of them arrive in a Poisson process with an average arrival rate of λi*ρ and λi*(1 − ρ). The packet service time of switches is assumed to follow an exponential distribution, and the expected service time is denoted 1/μ1 and 1/μ2, respectively. The mean service time of packet-in messages in the controller is denoted 1/μc. This service time includes the transmission time from the switches to the controller. In other, to simplify this model, both controller and switches are powerful enough for the traffic in the network, and there is no limit on the queue capacity. We queue all the packets arriving at a switch in a single queue instead of a separate queue on each ingress port and all the packets are processed in order of arrival time. Moreover, we assume that when the first packet of a connection arrives at a switch, the controller installs a flow entry. After that, the remaining packets arrive to the switch and are forwarded directly. We also assume that all the switches in our model have the same service rate, and the packet-in messages arrive the switch following a Poisson process.

5.1 OpenFlow Switch Performance

The flow entry matching for all packets are assumed to be independent and the packet processing time can be supposed to follow an exponential distribution. With the assumptions above the performance of OpenFlow switches can be modeled as a M/H2/1 queue,

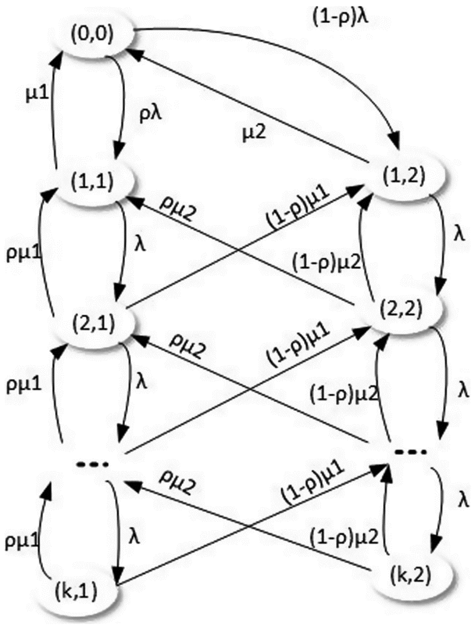

Fig. 9. State transition diagram of an M/H2/1 queue

which means packets arrive at the ith switch at rate λi and the service time is represented by a two-phase hyperexponential distribution. The state transition diagram of this queue is shown in Fig. 9. With probability ρ, a packet receives service at rate μ1, while with probability 1 − ρ it receives service at rate μ2.

A state is represented by a pair (a, b), where a is the total number of packets in the switch and b is the current service phase. In our case b can be only 1 or 2. The stationary distribution of this queue in the ith switch can be obtained by applying the Matrix-Geometric Method. We denote the stationary probability vector π(i) as

$$\pi^{(i)} = (\pi_0^{(i)}, \pi_1^{(i)}, \pi_2^{(i)}, \ldots, \pi_{k,}^{(i)} \ldots) \tag{1}$$

$$\rho = \frac{\lambda}{\mu} < 1 \tag{2}$$

$$\pi_0 = 1 - \rho \tag{3}$$

$$\pi_k = (1 - \rho)\rho^k \tag{4}$$

Where πk(i) is the probability of k packets in the ith switch.

Then the mean number of packets in the queueing system can be computed as:

$$N_i = \sum_{k=0}^{\infty} k\pi_k^{(i)} \tag{5}$$

$$N_i = \sum_{k=0}^{\infty} (1 - \rho)\rho^k \tag{6}$$

For k = 0, the product is zero then we can start the sum from k = 1

$$N_i = (1 - \rho) \sum_{k=1}^{\infty} k\rho^k = (1 - \rho)\rho \sum_{k=1}^{\infty} k\rho^{k-1} \tag{7}$$

Since $k\rho^{k-1}$ can be written as $k\rho^{k-1} = \frac{d\rho^k}{d\rho}$

Respectively

$$\begin{aligned} N_i &= (1 - \rho)\rho \sum_{k=1}^{\infty} \frac{d}{d\rho}\rho^k \\ &= (1 - \rho)\rho \frac{d}{d\rho}\left(\sum_{k=1}^{\infty} \rho^k\right) \end{aligned} \tag{8}$$

Since $\sum_{k=1}^{\infty} \rho^k = \sum_{k=0}^{\infty} \rho^k - 1 = \frac{1}{1 - \rho} - 1 = \frac{\rho}{1 - \rho}$

$$\text{We can write } N_i = (1 - \rho)\rho \frac{d}{d\rho}\left(\frac{\rho}{1 - \rho}\right) \tag{9}$$

$$N_i = \frac{\rho}{1-\rho}, \tag{10}$$

where $(\rho < 1)$ and $\rho = \frac{\lambda}{\mu}$

$$N_i = \frac{\lambda}{\mu - \lambda} \tag{11}$$

According to Little's law, the average packet processing time in the ith switch can be given by

$$W_{si} = \frac{1}{\lambda} N_i = \frac{1}{\mu - \lambda} \tag{12}$$

The mean packet processing time of switches can be given by

$$W_s = \sum\nolimits_{i=1}^{n} \frac{\lambda_i}{\sum\nolimits_{i=1}^{n} \lambda_i} W_{si} \tag{13}$$

5.2 Numerical Evaluation and Results

With the mentioned analytical framework and presented outcomes, we can evaluate the proposed queuing model with different parameters and report the upper bound of packet processing delay in the SDN switch. The switch average packet processing time is shown

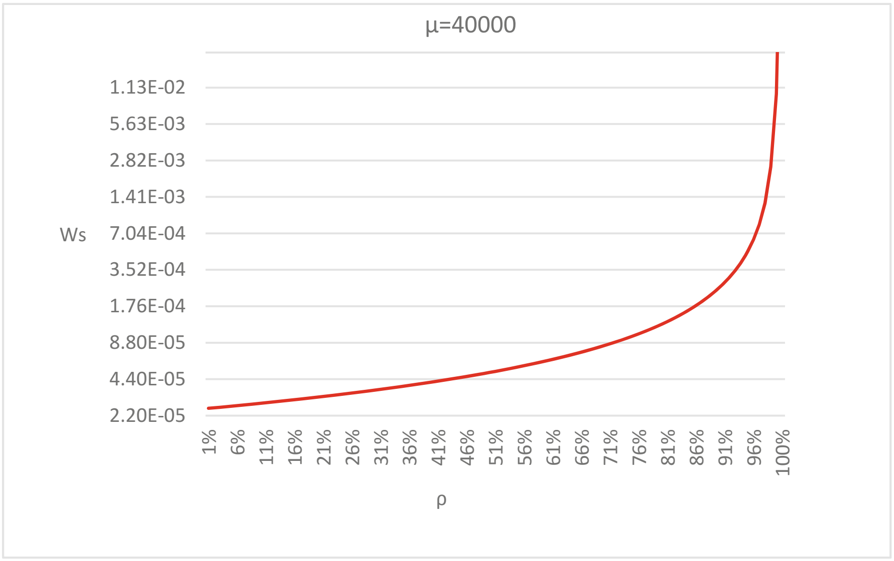

Fig. 10. Average packet processing time of switch

in Fig. 10. As packets arrival rate at the switch increases, the average packet processing time constantly increases. It sharply increases to the maximum when packet arrival rate is closer to the switch processing service rate. That matches the time when the switch runs out of resources and can't perform packet processing service.

6 Conclusion

Understanding the performances and limitations of OpenFlow-based SDN is a prerequisite of its deployment. In this work we have proposed a model for an OpenFlow SDN based on queueing theory, and resolves its average packet processing time. We reviewed the optimal parameter combinations of Openflow switch and controller to allow future network architects and administrators to be able to compute an upper bound estimation of packet delay and buffer requirement of SDN switches and controller for a given packet arrival rate.

Furthermore, we will extend the analysis to the possibility of SDN network from a single controller to the case of controller clusters to evaluate how much switches a given controller can handle in a network without much performance loss.

Acknowledgment. The publication was financially supported by the Ministry of Education and Science of the Russian Federation (the Agreement number 02.a03.21.0008), RFBR according to the research project No. 17-57-80102 "Small Medium-sized Enterprise Data Analytics in Real Time for Smart Cities Applications".

References

1. Kreutz, D., Ramos, F., Verissimo, P., Rothenberg, C., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015)
2. Xia, W., Wen, Y., Foh, C., Niyato, D., Xie, H.: A survey on software-defined networking. IEEE Commun. Surv. Tutorials **17**(1), 27–51 (2015)
3. Alsmadi, I.M., AlAzzam, I., Akour, M.: A systematic literature review on software-defined networking. In: Alsmadi, I.M., Karabatis, G., AlEroud, A. (eds.) Information Fusion for Cyber-Security Analytics. SCI, vol. 691, pp. 333–369. Springer, Cham (2017). doi:10.1007/978-3-319-44257-0_14
4. Liu, L., Zhang, D., Tsuritani, T., et al.: Field trial of an openflow-based unified control plane for multilayer multigranularity optical switching networks. J. Lightwave Technol. **31**(4), 506–514 (2013)
5. Jararweh, Y., Al-Ayyoub, M., Darabseh, A., Benkhelifa, E., Vouk, M., Rindos, A.: SDIoT: a software defined based internet of things framework. J. Ambient Intell. Humanized Comput. **6**(4), 453–461 (2015)
6. Kirichek, R., Vladyko, A., Paramonov, A., Koucheryavy, A.: Software-defined architecture for flying ubiquitous sensor networking. In: 19th International Conference on Advanced Communication Technology (ICACT), pp. 158–162 (2017)
7. OpenFlow. https://www.opennetworking.org/sdn-resources/openflow
8. Kirichek, R., Vladyko, A., Zakharov, M., Koucheryavy, A.: Model networks for internet of things and SDN. In: 18th International Conference on Advanced Communication Technology (ICACT), pp. 76–79. IEEE (2016)

9. Vladyko, A., Muthanna, A., Kirichek, R.: Comprehensive SDN testing based on model network. In: Galinina, O., Balandin, S., Koucheryavy, Y. (eds.) NEW2AN/ruSMART -2016. LNCS, vol. 9870, pp. 539–549. Springer, Cham (2016). doi:10.1007/978-3-319-46301-8_45
10. Vladyko, A., Letenko, I., Lezhepekov, A., Buinevich, M.: Fuzzy model of dynamic traffic management in software-defined mobile networks. In: Galinina, O., Balandin, S., Koucheryavy, Y. (eds.) NEW2AN/ruSMART -2016. LNCS, vol. 9870, pp. 561–570. Springer, Cham (2016). doi:10.1007/978-3-319-46301-8_47
11. Borshchev, A.: The Big Book of Simulation Modeling: Multimethod Modeling with Anylogic 6. AnyLogic North America (2013)
12. Dombacher, C.: Queueing Models for Call Centres. BDD (2010)
13. Xiong, B., Yang, K., Zhao, J., Li, W., Li, K.: Performance evaluation of OpenFlow-based software-defined networks based on queueing model. Comput. Netw. **102**, 174–183 (2016)
14. Xiong, B., Peng, X., Zhao, J.: A concise queuing model for controller performance in software-defined networks. J. Comput. **11**(3), 232–237 (2016)
15. Ansell, J., Seah, W., Ng, B., Marshall, S.: Making queueing theory more palatable to SDN/OpenFlow-based network practitioners. In: IEEE/IFIP Network Operations and Management Symposium (NOMS), pp. 1119–1124 (2016)