Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Evaluation of Controller Capacity in Software-Defined-Network with Batch Flow Requests

Ankita Jaiswal[1,2], Veena Goswami[3], Bibhuti Bhusan Dash[3], Utpal Chandra De[3], Suchismita Rout[1], Sudhansu Shekhar Patra[3*]

[1]School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar, India
[2]School of Computer Sc. & Engineering, Ramdeobaba University, Nagpur, India
[3]School of Computer Applications, KIIT Deemed to be University, Bhubaneswar, India
Email: sudhanshupatra@gmail.com

The development of SDN gives a new paradigm to networking by flow-based OpenFlow architecture, which decouples the control plane from the data plane. The controller manages the switches in the network, and the central issue in this network architecture is a controller manage how many switches. This paper investigates the controller capacity regarding the flow requests to the switches forwarded from the controller in batches and models the queueing system as Mx/M/1. We present the expressions in explicit form and derive the performance measures of the controller, such as the probability distribution of the system, cumulative distribution system, mean length of the system, the waiting time of the flow requests in the system, and utilization of the controller. These measures help the service providers to make the system model depending on the number of flow requests that arrive at the controller from the switches in an idle condition.
Keywords: SDN, Single-Controller Queuing Model, Batch Arrival, Mx/M/1, Openflow.

1. Introduction

SDN is considered to be the next-generation network intelligent system controlled centrally by the controller where control and data planes are separated from each other in comparision to the traditional network (Nunes et al., 2014; Feamster et al., 2014; Trois et al.,2016). In recent days SDN has shown rapid growth due to its centralized control management of network operations and the network is easily scalable. In data center infrastructures SDN is considered to be its building block as it is scalable in nature, highly efficient, and has a

virtualization feature. SDN was introduced due to poor network traffic management, scalability issues in the network, virtualization limitations, and enforcement in policy management. The traffic forwarding work is handled by the data plane and the intelligent decisions are handled by the control plane. In SDN, the controller assumes programmatic management of the whole network and enforces regulations for the underlying devices. SDN switches are tasked with routing traffic according to the directives provided by the controller (Abderrahmane et al., 2024) . The SDN architecture is composed of 3 layers and is shown in fig. 1:

Control Layer

In SDN, thisl layer is situated in between the Application layer and the Infrastructure layer. The SDN controller is placed in this layer which determines the flow of the traffic in the network.

Infrastructure Layer

All network devices, including switches and routers, are situated at this layer, which constitutes the foundational tier of the architecture and is tasked with directing traffic to the control layer.

Application Layer

The uppermost layer is accountable for relaying network requests via various APIs according to user requirements.

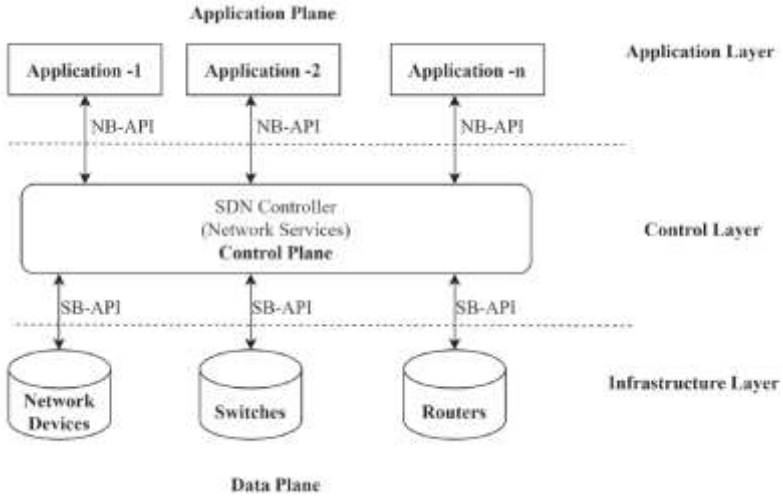

Fig. 1. SDN Architecture (Das et al., 2024)

OpenFlow (Miguel-Alonso, 2022), an open-source standard endorsed by several companies, is the first software-defined networking (SDN) control protocol. Openflow is a network control protocol. Network traffic does not traverse the OpenFlow protocol. OpenFlow transmits control signals instructing network switches on how to direct network traffic. In conventional network architecture, each switch has a routing table that it utilizes to determine the routing of each packet. This routing table is mostly static; it will be modified

by the administrator on each router separately. In OpenFlow, the SDN controller functions as the control plane. The controller encompasses the logic and executes the decision-making about the flow of network traffic among the switches. The SDN controller initiates a connection with each switch to transmit messages (Shirmarz et al., 2020; Yao et al., 2014; Yeganeh et al., 2013).

Fig. 2 depicts the OpenFlow network that this research theoretically analyses. In this setup, several OpenFlow switches are either directly or indirectly linked to the controller. Fig. 3 depicts the OpenFlow packet forwarding mechanism. A packet's header is examined by a switch upon receipt, and the results are compared to the flow table. If the header field wildcard is matching with an item in the flow table, then the entry is considered appropriate. In cases when many matching are found, matching of the packets is done based on priority, which means that the most specific entry or the most important wildcard is selected. Then the counters linked to that item in the flow table are changed by the switch. In the end, the packet's flow table entry specifies the actions that the switch should do, such as sending the packet to a certain port. Usually, when a packet's header doesn't match anything in the flow table, the switch will let its controller know.

OpenFlow controller may execute 2 operational epitomes: reactive and proactive. In the first mode, the controller passively monitors switches and sets up the flow table as needed. In this framework, the flow tables inside switches are updated in real-time. Upon the expiration of flow entry lifetimes, they will be removed from the flow table. In the second mode, the controller comprehends network information and preemptively populates the flow table. This will result in a reduced number of flow requests originating from switches. Consequently, the second mode reduces the flow forwarding latency in comparison to the first one. Nevertheless, the proactive mode cannot accurately represent changes in the network state in a timely manner. Fortunately, both the modes can operate together.

By modeling the flow set-up requests from switches to the controller as a batch arrival process $M^x/M/1$, we derive a closed-form formula for various performance measures of these requests. By considering the flow service time as a constraint, the number of switches manageable by a controller is established, hence offering a means to assess the controller's capability. We anticipate that the outcome will significantly enhance large-scale OpenFlow network implementation.

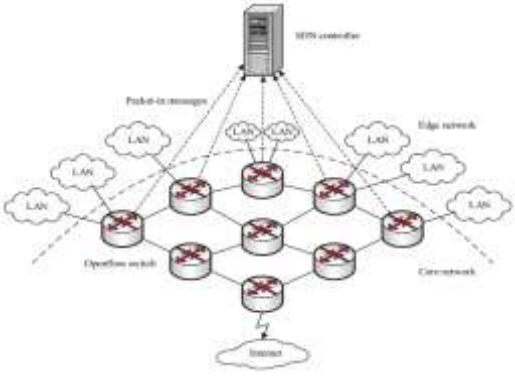

Fig. 2. Openflow Network

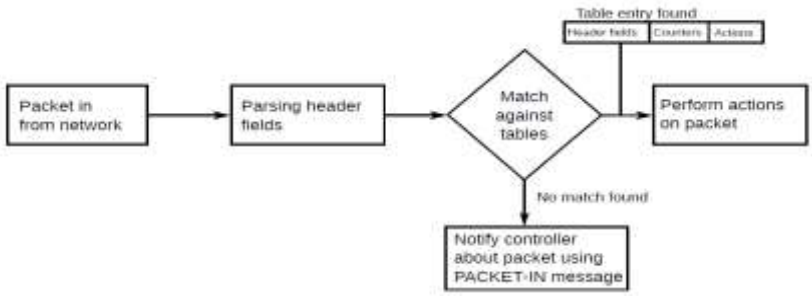

Fig. 3. Packet forwarding mechanism in Openflow Network (Braun et al., 2014)

A. Motivation

During the transmission of the packets in the network if a switch receives a new traffic flow request, the first packet is sent to the controller. It examines the header of the incoming packet, identifies the traffic flow route using topological data, and then sets up the destination and source data plane devices' forwarding tables. If many flow requests arrive simultaneously, the controller will not able to handle them and there will be controller disrupts in the overall network traffic and stops the process of setting up the flow. The number of switches a controller can handle is dictated by the constraints of a finite flow service time, hence offering a means to assess the controller's capacity.

B. Contributions

The following are the paper's main contributions:

- It models the system as batch arrival $M^x/M/1$ model where the flow requests are received from switches to the controller.
- It provides the performance of the single controller based on the flow of requests from switches to controllers.
- It can be extended further to a multi-controller from a single controller.

Various analytical studies have been made to validate the proposed system with numerical results. To visualize the consequences of different parameters, the results are shown as graphs and tables that illustrate the performance of the single controller while flow requests reached to the controller in batches from the switches.

C. Organization

The remainder of the paper is arranged as follows. Section 2 gives the previous work in the said area, Section 3 and 4 provides a description of the model and performance measures of the controller respectively. The results and discussion is depicted in section 5. At the end, the conclusion is depicted in Section 6.

2. Related Work

One of the most well-known protocols for software-defined networking (SDN) controller-to-switch connections is OpenFlow, which is often considered a potential strategy for the future

of the internet. Deploying SDN based on OpenFlow requires an awareness of its performance and restrictions.. The authors suggested (Xiong et al., 2017) a unique analytical performance evaluation strategy of OpenFlow networks established on queueing model. Following the illustration of a standard network scenario with OpenFlow installations, they represented the packet forwarding of OpenFlow switches and the packet-in message processing of the SDN controller as the queuing models $M^X/M/1$ and M/G/1, respectively. The model then developed a queueing system for OpenFlow networks focused on packet forwarding performance, and derive its closed-form equation for mean packet sojourn time along with the associated probability density function. Ultimately, the numerical analysis is conducted to assess the suggested performance model using various parameter values. The paper by Muhizi et al. (2017) introduces an OpenFlow-SDN-based network visualisation and performance assessment model that aids in network design and planning by assessing the impact of varying traffic loads and network utilisation on performance. To attain the desired objective, they used the AnyLogic Multimethod simulation program as a study methodology. This is a pioneering instance where the performance assessment of Software-Defined Networking (SDN) is conducted using queuing model simulation to assess variations in average packet processing time across different network parameters. This work's provided SDN model enables network managers and planners to more accurately anticipate performance changes resulting from traffic fluctuations. This enables them to make timely choices to prevent minor concerns from escalating into significant delays.

The paper (Jassar, 2018) proposes an analytical method for evaluating the QoS in SDN. SDN is a construct that intended to improve the performance and efficacy of computer networks. The present complexity of networks based on this notion necessitates the design of appropriate techniques for calculating task characteristics. The model defined Poisson arrival process and service time by a hyper-exponential distribution for QoS study. The performance analysis of the suggested queuing model demonstrates its accuracy. The QoS attributes calculated include the maximum flow intensity switch and controller can handle, the mean packet sojourn duration on individual serving components, as well as the delay and CPU utilisation of these serving elements.

The authors (Jarschel et al., 2011) developed a fundamental model for the forwarding speed and blocking probability of a switch integrated with a controller, based on observations of switching times from existing OpenFlow hardware, and validated it via simulation. This model is applicable to predict the packet sojourn time and the chance of lost packets in such a system and can offer insights to developers and researchers on questions how an OpenFlow architecture would behave given particular parameters. Although the model offers valuable insights, it is only accurate when the likelihood of anticipating a new flow is minimal. Secondly, extending the model to include multiple forwarding elements in the data plane is not straightforward. In this study, the authors (Mahmood et al., 2014) suggested a model that tackles both challenges. The model is founded on the Jackson assumption, with modifications specifically designed for the OpenFlow-based SDN network. The performance examination of the proposed model demonstrates its accuracy, even in scenarios when the likelihood of fresh flow is quite high. However, the literature is deficient on the scenario involving several nodes in the data plane. This study (Mahmood et al., 2015) presents a model to tackle this difficulty by estimating the data

plane as an open Jackson network, with the controller being represented as an M/M/1 queueing model. The model then analyzed the critical metrics, including the average duration a packet resides in an OF-based network and the volume of data that may be sent inside the network while adhering to average delay constraints. The PDF and CDF of the time a packet spends for a certain route have been obtained.

3. Model Description

Here, we assume MX/M/1 queuing system (Shortle et al., 2018; Medjhi, 2002), where the arrival of flow requests is in batches of size X from the switches to the controller is a random variable with probability function P(X=i)= a_i , i >= 1 and mean batch size $E(X) = \bar{a}$. The arrival process of flow requests follows Poisson process having mean $1/\lambda_h$. Assume the processing time of the controller follows exponential distribution with mean $1/\mu_h$. The buffer space is infinite, and service follows a FCFS discipline. The utilization factor is $\rho = \frac{\lambda_h \bar{a}}{\mu_h}$. The transition diagram is shown in fig. 4.

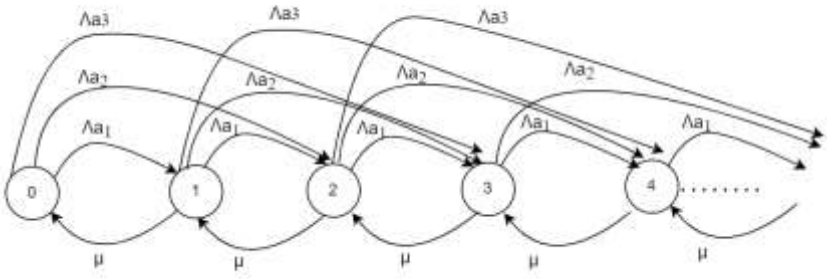

Fig. 4 . State transition diagram

Let us define the steady-state probabilities of the system by $Q_l, l \geq 0$. At the steady-state balance equations can be represented as

$$\begin{cases} \lambda_h Q_0 = \mu_h Q_1, & l = 0 \\ (\lambda_h + \mu_h) Q_l = \mu_h Q_{l+1} + \frac{\lambda_h}{\mu_h} \sum_{j=1}^{l} a_j Q_{l-j}, & l \geq 1 \end{cases} \quad (1)$$

Solving recursively (1), we have

$$Q_1 = \frac{\lambda_h}{\mu_h} Q_0 \quad (2)$$

$$Q_{l+1} = \frac{\lambda_h + \mu_h}{\mu_h} Q_l - \frac{\lambda_h}{\mu_h} \sum_{j=1}^{l} a_j Q_{l-j}, l \geq 1 \qquad (3)$$

Applying normalization condition, we can find only unknown Q_0 as $Q_0 = \left[1 + \sum_{j=1}^{\infty} Q_j\right]^{-1}$

The generating function $Q(z) = \sum_{l=0}^{\infty} z^l Q_l$ given in (1) as

$$Q(z) = \frac{\mu_h (1-z) Q_0}{\mu_h (1-z) - \lambda_h z (1 - A(z))} \qquad (4)$$

Where $A(z) = \sum_{j=1}^{\infty} a_j z^j . As\, Q(1) = 1,$ using L'Hospital's rule

$$Q_0 = 1 - \frac{\lambda_h \bar{a}}{\mu_h} = 1 - \rho .$$

Thus from Eq (5), we have

$$Q(z) = \frac{\mu_h (1-z)(1-\rho)}{\mu_h (1-z) - \lambda_h z (1 - A(z))} \qquad (5)$$

Specific distributions of batch sizes

Geometric distribution

If batch size assumes a geometric distribution, then $a_j = d(1-d)^{j-1}, j \geq 1, 0 \leq d < 1,$ the probability generating function $A(z) = \frac{dz}{1-(1-d)z}$, and mean batch size $\bar{a} = 1/d$. Thus we have

$$Q_l = \begin{cases} 1-\rho, & l = 0 \\ d\rho(1-\rho)[1-d(1-\rho)]^{l-1}, l \geq 1, \end{cases}$$

where $\rho = \lambda_h / (d\mu_h).$

Deterministic distribution

If batch size takes a deterministic distribution, then , the probability generating function $A(z) = z^j$, and mean batch size $\bar{a} = j$. We obtain

$$Q(z) = \frac{\mu_h(1-z)(1-\rho)}{\mu_h(1-z) - \lambda_h z(1-z^j)}$$

where $\rho = (j\lambda_h)/\mu_h$.

Truncated Poisson distribution

If the batch size assumes a truncated Poisson distribution, then $a_j = \frac{e^{-\beta}\beta^{j-1}}{(j-1)!}$, the probability generating function $A(z) = ze^{-\beta(1-z)}$, and the mean batch size $\bar{a} = 1+\beta$

4. Performance Measures

Performance measures, which give the effectiveness of the model being discussed, are essential elements of queueing systems. The system's performance measures may be given as follows:

- The average number of requests in the queue is

$$L_q = \sum_{i=1}^{\infty}(i-1)Q_i$$

- The average number of requests in the system is

$$L_s = \sum_{j=1}^{\infty} jQ_j$$

- The average waiting times for requests in the system (Ws) and the queue (Wq), respectively, are found using Little's law as

$$W_s = \frac{L_s}{\lambda\bar{a}} \text{ and } W_q = \frac{L_q}{\lambda\bar{a}}$$

5. Numerical Results

Graphs are used to display numerical validations in this section. By using the numerical examples below to highlight the qualitative elements of the queueing system under consideration, it assists managers in making informed decisions. Fig. 5 shows the probabilities of several requests for various distributions of fixed mean batch size (MBS). As the number of requests increases, probabilities decrease. We note that the probabilities of the number of requests are minimum in the case of geometric distribution and maximum in the case of deterministic distribution. Fig. 6 illustrates the CDF of the number of requests for various distributions of fixed mean batch size. The CDF increases with the increase in the

number of flow requests to the controller, and ultimately, it reaches the maximum value. As expected, the number of flow requests with the controller is maximum in the case of deterministic distribution batch size.

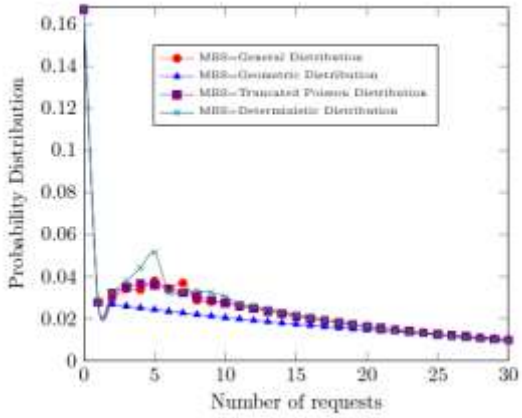

Fig. 5. Probabilities for various distributions of fixed mean batch size

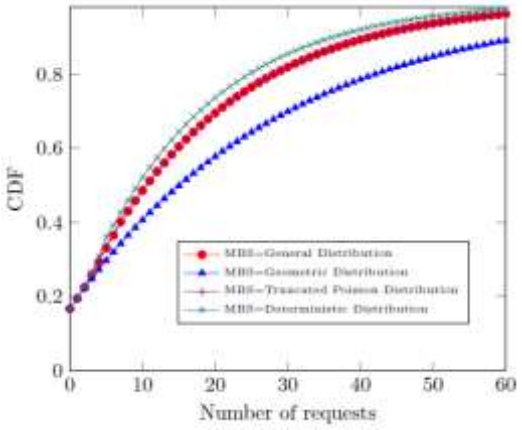

Fig. 6. The CDF for various distributions of fixed mean batch size

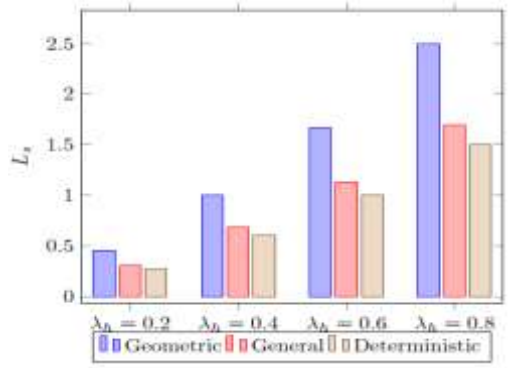

Fig. 7. Mean system length for various batch distributions Vs Arrival rate

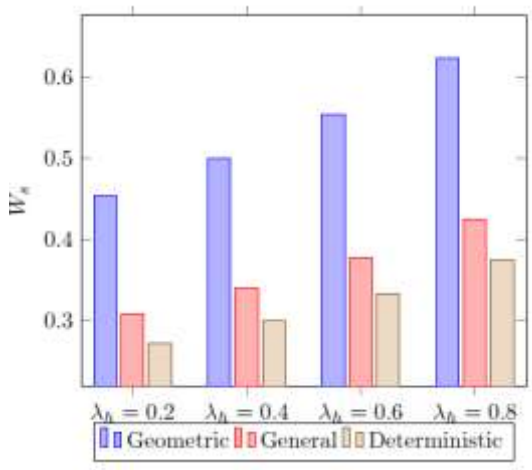

Fig. 8. Waiting time for various batch distributions Vs Arrival rate

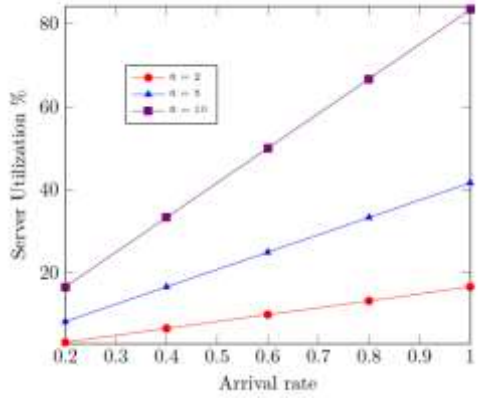

Fig. 9. Server utilization Vs Arrival rate

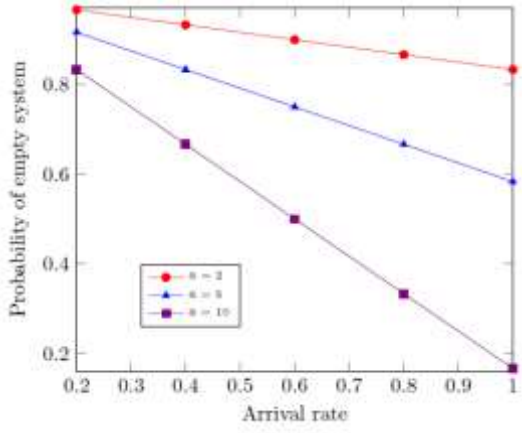

Fig. 10. Probability of empty system Vs Arrival rate

Fig. 5 plots the impact of arrival rate on the mean system length for various batch distributions. It is evident that as the arrival rate increases, Ls increases monotonically. For fixed λ, we see that for the same mean batch size, Ls is less in deterministic distribution and higher in geometric distribution. Fig. 6 depicts the effect of arrival rate on the mean waiting time for several batch distributions. The mean waiting time in the system increases with the arrival rate. Here, Ws is also less in deterministic distribution and higher in geometric distribution. Figs. 7 and 8 depict the effect of mean batch size on the server utilization and probability of an empty system, respectively. From fig. 9, we observe that server utilization increases with arrival rate for fixed mean batch size. Also, server utilization rises as the mean batch size increases for a fixed arrival rate. In Fig. 10, we see that probability of an empty system decrease with the increase of arrival rate under different mean batch sizes. For a fixed arrival rate, the probability of an empty system also decreases with the increase in mean batch size.

6. Conclusion

In this paper, we have developed steady-state solutions for the M[x]/M/1 queueing system for the flow requests arrival from the switches to the controller. Some important performance measures are also derived for the controller for various batch distributions. The results are shown in the form of graphs. The system is modeled for a single controller. But in real practice, there may be a multi-controller system where multiple controllers support the flow requests that arrive from the switches.

References

1. Abderrahmane, A., Drid, H., & Behaz, A. (2024). A Survey of Controller Placement Problem in SDN-IoT Network. International Journal of Networked and Distributed Computing, 1-15.
2. Braun, W., & Menth, M. (2014). Software-defined networking using OpenFlow: Protocols, applications and architectural design choices. Future Internet, 6(2), 302-336.
3. Das, D., Sahoo, B., Roy, S., & Mohanty, S. (2024). Performance Analysis of an OpenFlow-Enabled Network with POX, Ryu, and ODL Controllers. IETE Journal of Research, 1-18.
4. Feamster, N., Rexford, J., & Zegura, E. (2014). The road to SDN: an intellectual history of programmable networks. ACM SIGCOMM Computer Communication Review, 44(2), 87-98.
5. Jarschel, M., Oechsner, S., Schlosser, D., Pries, R., Goll, S., & Tran-Gia, P. (2011). Modeling and performance evaluation of an OpenFlow architecture. In 2011 23rd International Teletraffic Congress (ITC) (pp. 1-7). IEEE.
6. Jassar, A. A. (2018). An analysis of QoS in SDN-based network by queuing model. Telecommunications and Radio Engineering, 77(4).
7. Mahmood, K., Chilwan, A., Østerbø, O. N., & Jarschel, M. (2014). On the modeling of openflow-based sdns: The single node case. arXiv preprint arXiv:1411.4733.
8. Mahmood, K., Chilwan, A., Østerbø, O., & Jarschel, M. (2015). Modelling of OpenFlow-based software-defined networks: the multiple node case. IET Networks, 4(5), 278-284.
9. Medhi, J. (2002). Stochastic models in queueing theory. Elsevier.
10. Miguel-Alonso, J. (2022). A research review of OpenFlow for datacenter networking. IEEE Access, 11, 770-786.
11. Muhizi, S., Shamshin, G., Muthanna, A., Kirichek, R., Vladyko, A., & Koucheryavy, A.

(2017). Analysis and performance evaluation of SDN queue model. In Wired/Wireless Internet Communications: 15th IFIP WG 6.2 International Conference, WWIC 2017, St. Petersburg, Russia, June 21–23, 2017, Proceedings 15 (pp. 26-37). Springer International Publishing.

12. Nunes, B. A. A., Mendonca, M., Nguyen, X. N., Obraczka, K., & Turletti, T. (2014). A survey of software-defined networking: Past, present, and future of programmable networks. IEEE Communications surveys & tutorials, 16(3), 1617-1634.
13. Shirmarz, A., & Ghaffari, A. (2020). Performance issues and solutions in SDN-based data center: a survey. The Journal of Supercomputing, 76(10), 7545-7593.
14. Shortle, J. F., Thompson, J. M., Gross, D., & Harris, C. M. (2018). Fundamentals of queueing theory (Vol. 399). John Wiley & Sons.
15. Trois, C., Del Fabro, M. D., de Bona, L. C., & Martinello, M. (2016). A survey on SDN programming languages: Toward a taxonomy. IEEE Communications Surveys & Tutorials, 18(4), 2687-2712.
16. Xiong, B., Yang, K., Zhao, J., Li, W., & Li, K. (2016). Performance evaluation of OpenFlow-based software-defined networks based on queueing model. Computer Networks, 102, 172-185.
17. Yao, L., Hong, P., & Zhou, W. (2014, August). Evaluating the controller capacity in software defined networking. In 2014 23rd International Conference on Computer Communication and Networks (ICCCN) (pp. 1-6). IEEE.
18. Yeganeh, S. H., Tootoonchian, A., & Ganjali, Y. (2013). On scalability of software-defined networking. IEEE Communications Magazine, 51(2), 136-141.