A Concise Queuing Model for Controller Performance in Software-Defined Networks

Bing Xiong[1*], Xia Peng[1], Jinyuan Zhao[2]

[1] School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China.

[2] School of Software, Central South University, Changsha 410075, China.

* Corresponding author. Tel.: 086-0731-85258463; email: xiongbing.csust@qq.com

Manuscript submitted June 10, 2015; accepted August 5, 2015.

doi: 10.17706/jcp.11.3.232-237

Abstract: Software-defined Networking (SDN) is commonly seen as a promising way towards future Internet. Understanding the performance and limitation of its logically centralized controllers is a prerequisite of SDN deployments. For this end, this paper proposes a concise queuing model for SDN controller performance. We first investigate that the input of a SDN controller is a hybrid Poisson stream of packet-in messages. Then, we analyze the message processing of a SDN controller to achieve a concise queuing model $M/M/1$ for its performance, and further derive its performance parameters from queuing theory. Finally, our proposed model is evaluated by measuring a well-known controller with the benchmark Cbench. Experimental results indicate that our model is a better approximation of the controller performance than the existing ones.

Key words: software-defined networking, SDN controllers, performance evaluation, queuing model.

1. Introduction

As a new network paradigm, Software-defined Networking (SDN) is currently seen as one of the promising approaches in the way towards future Internet [1]. It decouples the network control out of the forwarding devices, and allows for a separate controller entity that may change the forwarding rules in modern switches. The separation of the control logic from the forwarding devices greatly facilitates the deployment and operation of new services and enables SDN to react gracefully to the change of network demands and modifications to the substrate topology. These pave the way for a more flexible, programmable, and innovative networking, but raise the bottleneck of controller performance [2].

Understanding the performance and limitation of SDN controllers is a prerequisite for their usage in practical applications. Specifically, an initial estimate of the performance and requirement of a SDN controller is essential for network architects and designers. Although simulation studies and experimentation are among the widely used performance evaluation techniques, analytical modeling has its own benefits. A closed-form description of a networking architecture enables the designers to have a quick approximate estimate of the performance of their design, without the need to spend considerable time for simulation studies or expensive experimental setups.

To the best of our knowledge, there are very few work to analytically model and evaluate the performance of a SDN controller. Jarschel *et al*. introduced a flexible OpenFlow controller benchmark Cbench to carry out the performance measurements on a per-switch basis [3]. The Cbench was utilized to measure different performance aspects of 4 publicly-available OpenFlow controllers (NOX, NOX-MT, Beacon, and Maestro) [4].

Shalimov *et al.* further developed a new framework hcprobe to test more performance indexes including efficiency, scalability, reliability, security [5] Close to our work, Azodolmolky *et al*. presented a mathematical framework based on network calculus to report the performance of the scalable SDN deployment [6]. Bozakov *et al.* used a queuing model to characterize the behavior of the control interface between the controller and a switch, and proposed a simple interface extension for controller frameworks which enables operators to configure delay bounds for transmitted control message [7]. Different from them, we focus on the average performance of a SDN controller at its equilibrium state. Most close to our work, Jarschel *et al.* derived a basic queuing model for an OpenFlow architecture, which merely considered a simple network scenario where a SDN controller is connected with a single OpenFlow switch [8]. Zuo *et al.* further investigated the queuing delay of a SDN controller with different OpenFlow network sizes by introducing a batch-arrival single-departure queue model [9]. However, the model did not properly characterize the arrival process of flow request messages at the controller.

With the above motivations, this paper aims to provide a more accurate performance model of a SDN controller in charge of multiple OpenFlow switches. To achieve this, we investigate the arrival process of flow request messages at the controller and the packet-in message processing of the controller. On this basis, we reach an appropriate performance model and derive its performance parameters based on queuing theory. The model is evaluated by the measurement of a popular SDN controller with the Cbench. By this way, we try to provide a guideline for network architects and designers to deploy SDN controllers.

The rest of this paper is organized as follows. In Section 3, we propose a performance model of a SDN controller chiefly based on the investigation of its message arrival pattern. Section 4 evaluates our proposed model with the controller performance benchmark Cbench. In Section 5, we conclude the paper.

2. Queuing Model of SDN Controllers

This section analyzes the arrival process of packet-in messages at a SDN controller, and proposes a concise queuing model for SDN controller performance.

2.1. Message Arrival Process

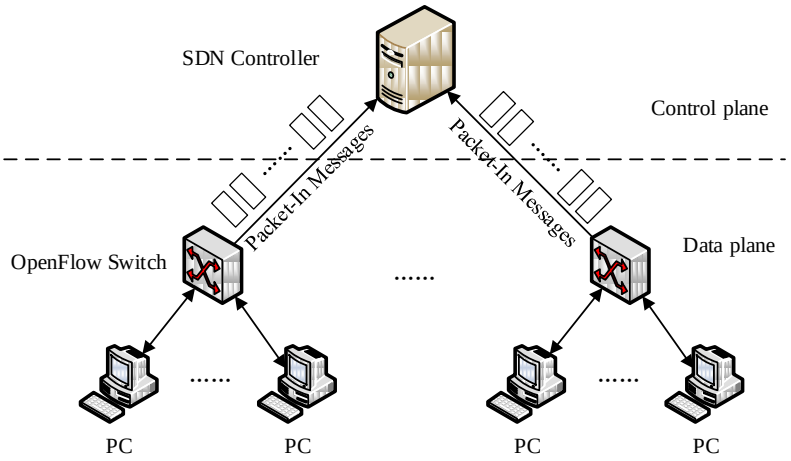


Fig. 1. A typical SDN architecture.

In SDN deployments, a controller usually manages multiple OpenFlow switches connecting a group of PC host. A typical SDN architecture is illustrated in Fig. 1 an OpenFlow switch performs flow table lookups on the arrival of a packet. 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 in a packet-in message to the upper SDN controller. Then the controller determines the corresponding flow rule and sends it in a packet-out or flow_mod message to the switch. As a consequence, a SDN controller receives a stream of packet-in messages from each OpenFlow switch.

Several studies have indicated that packet arrival process is not a Poisson stream due to the abruptness and persistence of network traffic, but flow emergence process conforms to Poisson distribution [10]. This means that the Poisson process is fit for the sequence of the first packets in all flows. In SDN architecture, an OpenFlow switch just sends the first packet of a flow in a packet-in message to its controller. Consequently, the packet-in messages from a switch to its controller conform to Poisson distribution, and the packet-in messages at a SDN controller is supposed to be a hybrid of multiple Poisson streams.

2.2. Controller Queuing Model

According to the above analysis, we can suppose that a SDN controller manipulates k switches and receives packet-in messages from the ith one in conformity to Poisson distribution with the parameter λ_i. Then, the message stream at the controller confirms to Poisson distribution with the parameter λ in (1) in terms of the composition theorem of Poisson stream. In particular, the parameter λ stands for the expectation of the arrival rate of the packet-in messages at the controller.

$$\lambda = \sum_{i=1}^{k} \lambda_i. \tag{1}$$

On the arrival of a packet-in message, the controller determines the respective flow rule by looking up its internal forwarding tables commonly called the forwarding information base (FIB). If the lookup fails, the controller resorts to its topology discovery and path computation to learn and process the message. Hence, the processing time of packet-in messages in the controller can be supposed to conform to negative exponential distribution with the parameter μ, which represents the expectation of the message processing rate. In summary, we can employ the concise queuing model $M/M/1$ to characterize the packet-in message processing of a SDN controller.

With the above queuing model, the length of packet-in message queue at a SDN controller $N(t)$ is a birth and death process on the countable infinite state set $\mathbf{E}$={0, 1, 2, …}. Then, we can express its instantaneous and stationary probability as follows.

$$p_j(t) = P\{N(t) = j\},\ p_j = \lim_{t\to\infty} p_j(t),\ j \in \mathrm{E}.$$

Suppose the traffic intensity of the controller is $\rho=\lambda/\mu$, we can derive the following conclusions from the limit theorems for birth and death processes: (a) If $\rho\geq 1$, $p_j=0$, $j\in \mathbf{E}$, does not constitute a probability distribution; (b) If $\rho<1$, $\{p_j,\in \mathbf{E}\}$ exists independent of initial condition, and $p_j=(1-\rho)\rho^j$, j=0, 1, 2…, form a geometric probability distribution. Hence, we can conclude the average queue length of the packet-in messages under the assumption of statistical equilibrium ($\rho<1$) in (2).

$$\overline{N} = E[N] = \sum_{j=0}^{\infty} j p_j = \frac{\rho}{1-\rho}. \tag{2}$$

Furthermore, we can also derive the average length of the waiting queue in (3).

$$\overline{N}_q = \sum_{j=1}^{\infty} j p_{j+1} = \frac{\rho^2}{1-\rho}. \tag{3}$$

According to the above queue length distributions, we can know that $p_0=1-\rho$ is also the probability of the controller in idle state, and ρ is just the probability of the idle controller in the busy state. Apparently, the controller grows busier with the bigger ρ.

Suppose the controller processes packet-in messages on a first-come/first-served basis, we can express

the distribution function of the waiting time of a packet-in message $W_q(t)$= P{$W_q \le t$} in (4).

$$W_q(t) = 1 - \rho e^{-(\mu-\lambda)t}, t \ge 0. \tag{4}$$

As a consequence, we can derive the average waiting time of a packet-in message in (5).

$$\overline{W_q} = E[W_q] = \frac{\rho}{\mu(1-\rho)}. \tag{5}$$

With the average waiting time, we can calculate the sojourn time of a packet-in message in the controller by adding its expected processing time in (6).

$$\overline{W} = \overline{W_q} + \frac{1}{\mu} = \frac{1}{\mu - \lambda}. \tag{6}$$

3. Performance Evaluation

This section evaluates the above model by measuring the performance of the well-known controller Floodlight with the publicly available benchmark Cbench [3].

3.1. Measurement Methodology

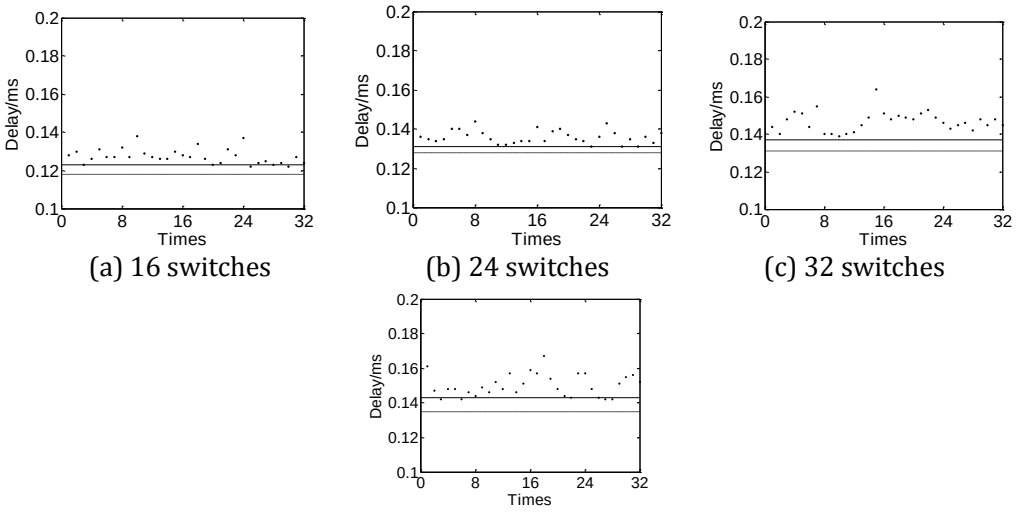
The tool Cbench is a prominent benchmark for SDN controller performance measurements. It is generally utilized to measure the performance of a SDN controller with the following steps: (a) simulating a network topology with a set number of OpenFlow switches and PC hosts; (b) generating a sequence of packet-in messages from each switch to the controller, and capturing their responses, i.e., packet-out or flow-mod messages; (c) recording the respective statistical information, and calculating the performance parameters of the controller.

The Cbench has two operation modes, i.e., throughput mode and latency mode. In throughput mode, each switch sends a continuous sequence of packet-in messages to overload the controller. So this mode can be utilized to measure the processing rate of a SDN controller for a given network scenario. In contrast, each switch won't send a packet-in message to the controller until it receives a response to the previous one in the latency mode. Thus, we can measure the sojourn time of a packet-in message and estimate the arrival rate of packet-in messages at the controller.

With the measurement results of the arrival rate of packet-in messages λ and the processing rate of the controller μ, we can compute an estimated sojourn time of a packet-in message in the controller according to (6). Finally, we evaluate our proposed model by verifying whether all measured sojourn time of packet-in messages falls around the estimated one in various network scenarios.

3.2. Performance Model Evaluation

We simulate 1024 unique MAC addresses per switch and different number of switches connected to the controller. The number of switches varies from 16 to 40 with a step length of 8. As for each scenario, we first measure the processing rate of the controller Floodlight in throughout mode. Then the sojourn time of a packet-in message in the controller is measured for 32 times in latency mode. The reciprocal of their average time is taken as the average arrival rate of packet-in messages to compute the estimated sojourn time of queuing models. Fig. 2 illustrates the measured sojourn time (scattered points) and the estimated ones of our proposed queuing model (horizontal solid line) and the multiple-arrival single-serving queue model [9] (horizontal dash line). As shown in Fig. 2, the estimated time of our proposed model is closer to the measured time than the multiple-arrival single-serving one. Therefore, we can conclude that our model is a better approximation of SDN controller performance.



(a) 16 switches (b) 24 switches (c) 32 switches

(d) 40 switches

Fig. 2. The estimated and measured sojourn time with different number of switches.

4. Conclusion

Understanding the performance and limitation of SDN controllers is a prerequisite for SDN deployments. This paper presents a queuing model for SDN controller performance. The model is evaluated by the measurement of a well-known controller Floodlight with the benchmark Cbench. Experimental results indicate that our proposed model is a good approximation of the controller performance.

Acknowledgment

This work was supported in part by National Natural Science Foundation of China (61303043, 61402055), Hunan Provincial Natural Science Foundation of China (2015JJ3010, 13JJ4052), and Scientific Research Fund of Hunan Provincial Education Department (15B009).

References

[1] McKeown, N., Anderson, T., Balakrishnan, H., *et al.* (2008). OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review (CCR)*, *38(2)*, 69-74.

[2] Benamrane, F., Mamoun, M. B., & Benaini, R. (2014). Short: A case study of the performance of an OpenFlow controller. *Proceedings of the 2nd International Conference on Lecture Notes in Computer Science* (pp. 330-334).

[3] Jarschel, M., Lehrieder, F., Magyari, Z., *et al.* (2012). A flexible OpenFlow-controller benchmark. *Proceedings of 1st European Workshop on Software Defined Networking* (pp. 48-53).

[4] Tootoonchian, A., Gorbunov, S., Ganjali, Y., *et al.* (2012). On controller performance in software-defined networks. *Proceedings of USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services* (pp. 1-6).

[5] Shalimov, A., Zuikov, D., Zimarina, D., *et al.* (2013). Advanced study of SDN/OpenFlow controllers. *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia* (pp. 1-4).

[6] Azodolmolky, S., Wieder, P., & Yahyapour, R. (2013). Performance evaluation of a scalable software-defined networking deployment. *Proceedings of the 2nd European Workshop on Software Defined Networks* (pp. 68-74).

[7] Bozakov, Z., & Rizk, A. (2013). Taming SDN controllers in Heterogeneous hardware environments. *Proceedings of the 2nd European Workshop on Software Defined Networks* (pp. 50-55).

[8] Jarschel, M., Oechsner, S., Schlosser, D., *et al.* (2011). Modeling and performance evaluation of an openflow architecture. *Proceedings of the 23rd International Teletraffic Congress* (pp. 1-7).

[9] Zuo, Q., Chen, M., & Jiang, P. (2013). Delay evaluation of OpenFlow control plane by queue model. *Journal Huazhong University of Science and Technology (Natural Science Edition)*, *8(1)*, 44-49.

[10] Williamson C. (2011). Internet traffic measurement. *Internet Computing*, *5(6),* 70-74.

Bing Xiong was born in August 1981, Yiyang, Hunan. He received his PhD in 2009 by master-doctorate program from the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), China, and the BSc in 2004 from Hubei Normal University, China. He is currently a lecturer in the School of Computer and Communication Engineering, Changsha University of Science and Technology, China. His main research interests include software defined networks, future internet architecture, network security.

Xia Peng was born in December 1990, Xiangtan, Hunan. She received her BS in 2013 from Changsha University of Science and Technology, China. She is currently a MS candidate in the School of Computer and Communication Engineering, Changsha University of Science and Technology, China. Her main research interests include software defined networks and network performance evaluation.

Jinyuan Zhao was born in January 1980, Shaoyang, Hunan. She received her MS and BS in 2007 and 2004, respectively, from Central China Normal University and Hubei Normal University. She is currently a lecturer in the Department of Computer and Communication, Hunan Institute of Engineering, China, and also a PhD candidate of software engineering in the School of Software, Central South University (CSU), China. Her main research interests include cloud computing and network security.