Evaluating the Controller Capacity in Software Defined Networking

Long Yao*, Peilin Hong*, Wei Zhou†
* The Information Network Lab of EEIS Department USTC, Hefei, China, 230027
yaolong@mail.ustc.edu.cn, plhong@ustc.edu.cn
† Huawei Technology Corporation Limited
will.zhou@huawei.com

***Abstract*—The flow-based OpenFlow architecture decouples the control plane and data plane, and it has involved great evolution towards traditional networks. A particular important issue in OpenFlow architecture is controller capacity, which can be defined as the number of switches a controller can manage. In this paper, we model the flow set-up requests from switches to controller as a batch arrival process $M^k/M/1$. Further, we analyze the controller performance with queuing theory, and derive the expression of average flow service time. Under the circumstance of a limited flow set-up time, the number of switches is determined, this provides a method to evaluate the controller capacity. Moreover, we extend the scene of a single controller to multiple controllers. All of these results are meaningful to large scale OpenFlow network deployment in the future.**

***Key Words*: OpenFlow, Controller, Batch-arrival, Capacity**

I. Introduction

The introduction of Software Defined Networking (SDN) [1] has caused an evolution towards traditional networks. The concept of SDN allows the separation of control plane and forwarding plane, namely moves the control logic to an external entity which is called controller. The most representative SDN realization is OpenFlow [2], which was first proposed by McKeown et al. in 2008 to enable researchers to conduct experiments in production networks.

OpenFlow has emerged as an important enabler of SDN. Comparing with traditional networks, OpenFlow network offers higher flexibility as it performs flow-based rather than packet-based switching. In OpenFlow architecture, OpenFlow is served as the communication protocol between software-based controller and OpenFlow switches. By separating the control plane from data plane, OpenFlow can satisfy fine grained flow processing and is programmable for researchers. The controller is responsible for determining the forwarding rules in switches. Therefore, the controller is often marked as the "Networking Operating System" in SDN. And OpenFlow switches are the forwarding devices which are controlled by one or more controllers. Whenever the switch receives a new flow, it requires the controller to install appropriate forwarding rules along the desired flow path.

It has been generally accepted that OpenFlow is an important criterion of southbound interface in SDN [3]. However, OpenFlow is still in its infant, and many aspects in OpenFlow remain to be improved [4]. An important issue to be addressed is the controller capacity, which can be defined as the number of switches a controller can manage in OpenFlow network.

In this paper, we focus on theoretical analysis of OpenFlow network depicted at Fig. 1, where many OpenFlow switches connect to the controller directly or via other switches. Modeling the flow set-up requests from switches to controller as a batch arrival process $M^k/M/1$, we obtain closed-form expression of the average service time of flow set-up requests. Under the circumstance of a limited flow service time, the number of switches that can be managed by a controller is determined, this provides a method to evaluate the controller capacity. We believe that the result will contribute a lot to large scale OpenFlow network deployment.

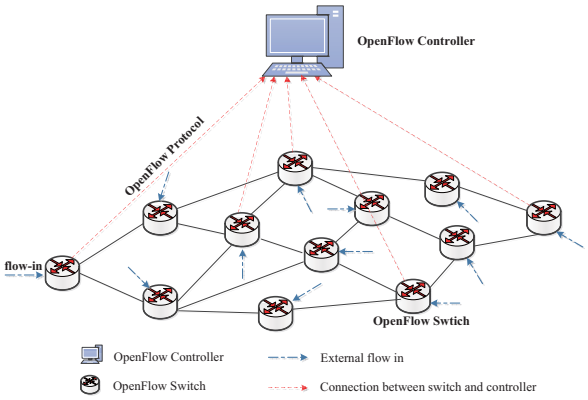

Fig. 1. OpenFlow Network

The main contributions of this paper are stated as follows:

- We model the flow set-up requests from switches to controller as a batch arrival process in queuing theory.
- We provide a method to evaluate the capacity of controller and present some in-depth discussions.
- We extend the scene of a single controller to multiple controllers.

The rest of paper is structured as follows. Section II gives a brief introduction to OpenFlow and states the related works. In section III, the system model is presented. Section IV is the analysis of the model. And Section V extends the model to multiple controllers. Finally a summary concludes this paper with an outlook of future works in section VI.

978-1-4799-3572-7/14/$31.00 ©2014 IEEE

II. Background and Related Work

The OpenFlow network has several components: OpenFlow controller, OpenFlow switches, and OpenFlow protocol. The centralized OpenFlow controller is responsible for configuring all OpenFlow switches it manages. The OpenFlow switches should be kept simple in order to achieve better forwarding performance. And OpenFlow protocol defines the messages interacted between controller and switches.

In OpenFlow network, each switch holds a flow table designed for flow matching. When a packet arrives at the switch, its header information is extracted and matched against the flow entries in switch. In the situation of a matched flow entry, the corresponding action is taken. Whereas if there is no matching entry in switch, the packet will be forwarded to controller (message "Packet-in"), then the controller will determine the rule to handle the packets (message "Packet-out") [3].

In our studies, we aim to evaluate the capacity of OpenFlow controller. Before that, we give a brief introduction to the operation paradigms of controller. An OpenFlow controller can perform two operation paradigms: reactive and proactive. In reactive mode, the controller listens to switches passively and configures the flow table on-demands. In this paradigm, the flow tables in switches update in time. When the lifetime of flow entries expire, they will be deleted from the flow table. While in proactive mode, the controller understands the information of network, it fills the flow table in advance. In this way, there will be less flow requests originate from switches. Therefore, the proactive mode decreases the flow forwarding delay compared with the reactive mode. However, the proactive paradigm can not reflect the network state changes timely. Fortunately, these two paradigms can work in parallel.

Currently, there have been a small number of studies towards the capacity of OpenFlow controller. Most of the researchers focus on verifying that the decoupling of control plane from forwarding plane is applicable in various areas, such as failure recovery in WAN [5], or traffic engineering in enterprise networks [6]. Bianco et al. [7] measured OpenFlow switching performance of different types of rules and packets sizes. However, the performance of switch-controller interaction had not been taken into consideration. Marcial et al. [8] and Tootoonchian et al. [9] evaluated the controller performance from the perspective of reactive and proactive paradigm of controller. Nevertheless, they did not give the theoretical results. Jarschel et al. [10] modeled the controller as a $M/M/1$ queuing system. The author considered the probability of unmatched packets when a new flow entered switches, and finally derived the total sojourn time of a packet through the OpenFlow network. However, the model only analyzed the situation where only one switch was connected to controller.

III. System Model

The common SDN implementation today relies on a centralized controller that possesses a global view of the network. An example of OpenFlow network is shown in Fig. 1, in which case the controller manages the access switches and core switches. All the switches connect to controller directly or via other switches.

We proceed in a straightforward way to model the OpenFlow network. Each access switch in the network will generate flow set-up requests. Therefore, during the service time of flow requests in controller, there will be several "packet-in" requests originating from switches to controller. We adopt the batch-arrival process $M^k/M/1$ in queuing theory to analyse the capacity and performance of controller.

A. Flow set-up process

To begin with, we describe the process of flow processing in OpenFlow controller. Whenever a switch receives a new flow, it requests the controller to install appropriate forwarding rules along the desired flow path. The time required to complete this operation is known as the flow setup time [11]. The flow set-up time is an important parameter in performance evaluation of OpenFlow architecture.

A schematic diagram is given in Fig. 2. Switch A receives a packet from host A without any flow entry matched, the default action of switch A is sending this packet to controller. Afterwards, the controller will determine the routing path of this flow, and send the corresponding flow entries to the alongside switches, in this case the alongside switches are switch A and B. After that, the packets belong to the same flow will go through switch A and B directly without requesting the controller.

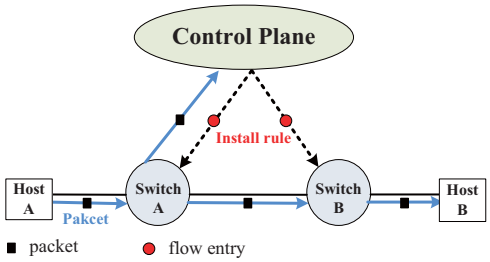

Fig. 2. Packet transmission with flow rule installation

To have an intensive understanding of the flow set-up process, the connection between switch and controller can be abstracted as a TCP connection. The flow set-up time equals double of propagation delay plus flow service time of controller. If we leave out the propagation delay between switches and controller, the flow set-up time is approximately the flow service time in controller.

B. Assumptions

The following assumptions are made before we describe the system model.

Assumption 1: The controller manages k access switches and some core switches. Each of access switches has at least a new arrival flow. When the new flow arrives, access switches will generate "Pakcet-in" message to controller, then

the controller will send the flow entry to the corresponding access switches[1] and core switches.

Assumption 2: The batch arrival instants of new flow set-up requests constitute a Poisson process with rate λ.

Assumption 3: The processing time of flow requests are independent, identically random variables with negative exponential distribution, mean $\frac{1}{\mu}$.

Assumption 4: The storage of controller is assumed to be infinite.

The first assumption is practical. The purpose of this model is to derive the number of switches that a controller can manage. Thus we should ensure the interaction between controller and switches. The switches are connected to controller directly or via other switches using specific algorithm like the shortest path first algorithm. In addition, each access switch has a new arrival flow guarantees that it will generate flow set-up requests to controller.

For the second assumption, two details should be claimed:

(1) In flow-based OpenFlow network, only the first packet of flow need to be sent to controller. Therefore, we only focus on the new flow set-up requests.
(2) The flow entries in switches are time-limited by idle time and hard-time [3]. Nevertheless, the assumption is also reasonable because we can regard the expired flow entries as new flow requests and they will go through the regular new flow set-up process.

The third assumption is based on the observation that the processing time of different flow set-up requests in controller is differ from each other. The time is strongly related to the processing rate of hardware and software versions of controller. Therefore, we provide the flow processing time as a random variable with exponential distribution. It should also be noted that the following analysis is applicable to a general batch-size distribution.

The fourth assumption is feasible with the development of storage devices. Therefore, the storage of requesting flows in controller is not the main problem. In other words, the controller is able to handle entire flow set-up requests, despite the service time.

Based on the assumptions above, we model the requests of flow entry from switches to controller as a batch arrival queuing system as shown in Fig. 3, where λ_i is the arrival rate of flow requests from switch i and μ is the average flow service time of OpenFlow controller. In addition, the arrival rate of flow requests satisfy that $\lambda_1 + \lambda_2 + \cdots + \lambda_n = k\lambda$. The flow requests between switches and controller will converge at the egress of controller, and form the queue. In following analysis, we focus on the theoretical analysis of controller performance with queuing theory, mainly address the service time of flow set-up requests and the queue size of controller.

C. Steady-state parameters of the model

As the queuing system of this model is Markovian, the steady-state probability is determined by a set of Chapman-Kolmogorov equations in steady state, using the standard queuing technology [12]. Considering the batch arrival process $M^k/M/1$, let

$$P_n = Pr\{\text{arriving batch encounters } n \text{ in system}\}, \quad (1)$$

$$g_i = Pr\{\text{arbitrary batch is of size } i;\ 1 \le i \le k\}, \quad (2)$$

and g_i satisfies poisson distribution [13], we have

$$g_i = Pr\{X = i\} = \frac{\lambda^i}{i!}e^{-\lambda}. \quad (3)$$

The state transition diagram is shown in Fig. 4.

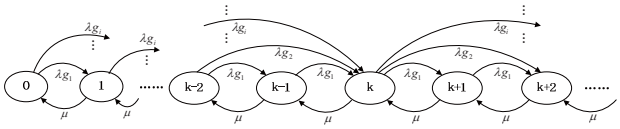

Fig. 4. $M^k/M/1$ State transition diagram

By inspection, the following Chapman-Kolmogorov equations can be hold.

$$\begin{cases} \lambda p_0 = \mu p_1 & k = 0 \\ (\lambda+\mu)p_k = \mu p_{k+1} + \sum_{i=0}^{k-1} p_i \lambda g_{k-i} & k \ge 1 \end{cases} \quad (4)$$

Applying the generation function [12], we could derive the steady state probability, the inversion of its Laplace transform can be performed numerically using the Matlab program. However, it is of a little complicated of the solution. Therefore, we search for a convenient method.

Suppose there are n flows waiting to be processed when the batch flow arrives, they should wait for the termination of previous n flows in the queue.

(1) During the processing time of recently batch-arrival flows, the average processing time of each flow request is

$$\begin{aligned} S &= \sum_{i=1}^{k} \frac{i}{\mu} Pr\{\text{the } i\text{th flow in } k\text{-batch arrival}\} \\ &= \sum_{i=1}^{k} \frac{i}{\mu} \cdot \frac{1}{k} \\ &= \frac{k+1}{2\mu}. \end{aligned} \quad (5)$$

Hence the average service time of each flow is

$$\begin{aligned} T &= \sum_{n=0}^{\infty} E\{\text{service time} \mid \text{arrival queue is } n\} \cdot p_n \\ &= \sum_{n=0}^{\infty} (\frac{n}{\mu} + \frac{k+1}{2\mu}) \cdot p_n \\ &= \frac{\lambda k \cdot T}{\mu} + \frac{k+1}{2\mu}. \end{aligned} \quad (6)$$

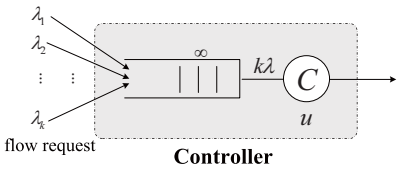

Fig. 3. The OpenFlow Queue Model

[1]For convenience, we refer switch as access switch hereinafter.

From front and end of eq.(7), we derive that

$$T = \frac{k+1}{2(\mu - \lambda k)} \quad (\mu > \lambda k). \tag{7}$$

(2) With Little's law [12], the average queue size of OpenFlow controller is

$$N = \lambda k \cdot T = \frac{\lambda k(k+1)}{2(\mu - \lambda k)} \quad (\mu > \lambda k). \tag{8}$$

For convenience, table I is listed to distinguish the symbols mentioned above.

TABLE I
THE SYMBOLS IN ANALYSIS OF CONTROLLER PERFORMANCE

Symbols	Definitions
k	The number of access switches
μ	Average flow processing rate of controller
λ	Average new flow arrival rate in each switch
S	Average flow processing time of controller
T	Average flow service time
N	Average queue size

According to previous derivation, we obtain the numerical results of average service time of flow set-up requests, and queue size of controller. It is discover from eq.(7) that average service time increases with number of switches. With the purpose of obtaining the number of switches a controller can manage, two issues should be concerned. Primarily, the switch number k is limited by flow processing rate λ and new flow arrival rate μ with constraint condition of $\lambda k < \mu$. Further, the number k is determined when the average flow service time T is limited. We ignore the details of queue size since we had assumed the storage of controller is infinite. Whereas, we can estimate the controller storage in demand from eq.(8). In following sections, we will focus on the analysis of flow service time.

IV. Analysis

It is obtained from eq.(7) that the decisive factors of average flow service time are flow processing rate of controller, arrival rates of new flows and number of switches. In this section, we focus on analyzing the average flow service time T in determine the controller capacity, and discussing the principles to choose T.

A. Numerical Simulation

To start with, we demonstrate the applicable of equations in previous analysis, and present the numerical simulations. An approximation of eq.(7) is performed when μ is considerably larger than λ,

$$T = \frac{k+1}{2(\mu - \lambda k)} = \frac{1}{2} \cdot \frac{1}{\frac{\mu+\lambda}{k+1} - \lambda} \approx \frac{1}{2} \cdot \frac{1}{\frac{\mu}{k+1} - \lambda} \tag{9}$$

It is even more evident from the approximation that average service time increases as switch number rises.

Then, the flow processing rate of controller and arrival rates of new flows are to be determined. There has been several researches towards them. Tavakoli et al. [15] revealed

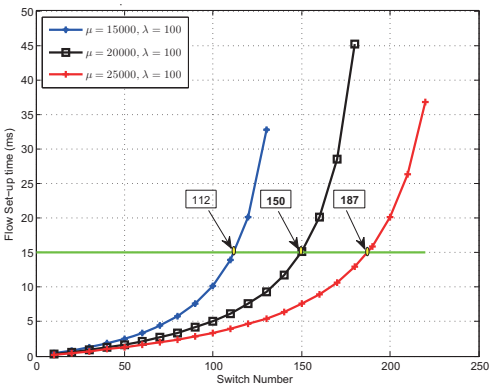

Fig. 5. The relationship under different service rates

that on currently deployed controllers, an individual controller could handle at least 30K new flow installs per second while maintaining a sub-10ms flow install time. Kandula et al. [16] indicated that in a cluster network with 1500 servers, there are about 100K flow requests generated, and the network requires at least 4 controllers.

According to the statistics above, we can reasonably suppose that the average new flow requests of each switch is no more than 100 per second, and the processing rate of controller is about 20K flow installs per second. In following numerical simulations, we adopt $\mu = 20K$ and $\lambda = 100$ as the baseline.

The relationship between average flow service time T and number k of switches is shown in the intermediate curve of Fig. 5, where flow arrival rate is λ =100 and service rate of controller is μ=20K. Obviously, this curve reveals the number of switches a controller can manage under a limited average flow service time. For instance, if the average flow service time T is restricted to 15ms, the maximum number k of switches a controller can manage is about 150. In other words, the capacity of controller is 150 switches maximize. In further, it reveals that when k reaches a specific threshold, T increases quickly, the scope is determined by the gradient of curve. As a consequence, the number k of switches should be limited under a certain bound to guarantee good performance.

The other curves at Fig. 5 show the relationships between flow service time and the number of switches under different flow processing rates of controller. Keep the flow arrival rate λ unchanged as 100, altering the flow processing rate μ as 15K, 20K, and 25K, respectively. Different values are adopted to simulate different processing rates of controller. As illustrated at Fig. 5, with a higher flow processing rate, the service time of a new flow request is lower. More directly, if the average flow service time is limited under $15ms$, the maximum number of switches are 112, 150, 187 respectively, as the boxes shown in Fig. 5.

Fig. 6 shows the relationship between average flow service time and number of switches under different new flow arrival rates. Keep the flow processing rate of controller as μ=20K, altering the flow arrival rates λ as 50, 80, 100, respectively. Different flow arrival rates are taken to demonstrate different

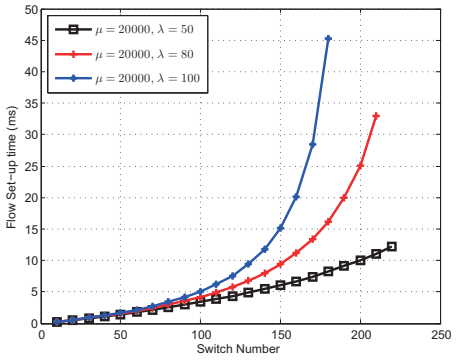

Fig. 6. The relationship under different flow-arrival rate

operation paradigms of controller, such as proactive, reactive or the hybrid [8]. When the controller performs as reactive paradigm, it configures the flow entries on demand. In this way, the new flow arrival rate is larger, result in a higher T. On the other hand, when acting as proactive paradigm, controller has the intelligence to understand the traffic behavior, thus new flow arrival rate is lower, result in a lower T. Obviously, the hybrid paradigm relies in between.

B. Flow service time

As shown in eq.(7), with the increasing of switch number k, the average flow service time T rises either. Therefore, if T is restricted within a bound, we can calculate the maximal number of k, and it equals the capacity of controller. In other word, average flow service time T determine the capacity of controller. Therefore, the flow service time T is of significant importance, we can state the following reasons:

(1) The flow service time determines the buffer size cached in switch. As OpenFlow switch specification describes, only the first packet of new flow will be forwarded to controller while the remaining parts of this flow are cached in switch [3]. If the flow service time is too large, the packets cached in switch will be exploded, and lead to the overflow of switch buffer.

(2) The service time of new flow installation will influence the end-to-end delay, and the delay is an important issue in QoS demands of different traffic models.

(3) The service time influences the consistence of flow updating in controller. In worst case, the communication node will regard it as a packet-loss event and send identical packets repeatedly if the flow service time is too long, thus the switch will generate duplicate flow requests to controller and cause the inconsistent flow update problem.

(4) For a given flow service time, the number of switches a controller can manage is determined by eq.(7).

C. How to determine the average flow service time ?

When determine the flow service time, two issues should be considered: QoS demands of different traffic models and the switch buffer size. We will discuss these two issues in detail.

(a) The QoS demands of different traffic models

Obviously, different traffic models have different QoS demands, especially in terms of end-to-end delay. We can get the demanded delay of different services from ITU standardization. In [14], the authors provide an indication of suitable performance targets of audio, video and data applications. Therefore, for different kinds of traffic models, the capacity of controller may vary diversely since the demands of flow service time are totally different, and the flow service time are determined by the minimum delay of traffics. Meantime, we should keep in mind that the primary limitation of controller capacity is $k < \mu/\lambda$.

Here we give a detailed example. Interactive applications like Voice over IP (VoIP) and Videophony require the value of delay of $100ms$ due to the desire of instantaneous feedback [14], thus the required one-way delay is limited to $50ms$. To achieve perfect transmission performance, we can reasonably suppose the average flow service time T is $15ms$. Concretely, we can get the capacity of controller is about 150 switches under the baseline conditions by eq.(7).

(b) The switch buffer size

Besides the QoS demands of different traffic models, flow service time is also determined by switch buffer size. As described before, if the flow service time is too large, the packets cached in switch will be exploded, this will lead to the overflow of switch buffer, and decrease the network performance.

In [17], the author revealed that switch buffer size is about 625000Byte in order to achieve 100% link utilization. And in [18], it described network bandwidth requirements for media traffics. We can calculate the maximum flow service time T with the following equation:

$$T \leq \frac{Buffer\_size}{\lambda \cdot R} \tag{10}$$

Suppose that the switch buffer size equals 625000Byte, the average data rates R refers to minimum video payload bitrate in order to guarantee a certain resolution, as depicted at table II. We adopt the average new flow arrival rates $\lambda = 100$ in each switch. In this way, we can calculate the maximum allowable average flow service time respectively. The results are given at table II.

TABLE II
VIDEO RESOLUTION BANDWIDTH

Video codec	**Resolution and aspect ratio**	R **(Kbps)**	T **(ms)**
$H.264$	848x480 (16:9)	400	125
$H.264$	960x540 (16:9)	500	100
$H.264$	1280x720 (16:9)	700	71.4
$H.264$	1920x1080 (16:9)	1500	33.3

With the increasing of new flow arrival rate, the allowable average flow service time is lower so as to guarantee perfect network performance, therefore the number of switches will decrease from eq.(7).

At first glance, the capacity of controller is related to hardware performance. However, we demonstrate that it also

depends on demands of flow service time which is determined by QoS demands of different traffic models and switch buffer size. In real OpenFlow networking deployment, these two issues should be considered simultaneously.

V. The model of multiple controllers

The common SDN implementation relies on a centralized controller that possesses a global view of the network. However, a single controller inevitable has the scalability problem [4]. Firstly, it is hard to find an optimal controller placement to ensure the acceptable latency between switches and controller. What's worse, a single controller has single-point-failure problem. Therefore, the architecture of multiple controllers in the form of cluster is proposed. In this section, we address the model of multiple controllers in SDN.

For convenience, the benefits of multiple controllers are summarized as follows:

- Scalability enabled.
- Enhance fault tolerance of controllers.
- Realize load balancing of controllers.
- Reduce the latency from switch to its closest controller.

In further, we analyze the performance of multiple controllers. Modeling the flow set-up requests from switches to controllers as a batch arrival process $M^k/M/m$ with m controllers, the state transition diagram is depicted at Fig. 7.

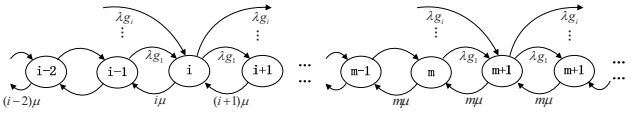

Fig. 7. $M^k/M/m$ state transition diagram

With the similar method we adopt in single controller, the average flow service time turns to

$$T = \frac{k+1}{2(m\mu - \lambda k)} \quad (m\mu > \lambda k) \tag{11}$$

in scenario of m controllers.

It is concluded that multiple controllers have better performance towards single controller, and the advantage mainly focuses on the flow processing rate. In other words, we can reduce the flow service time by deploying more controllers in the cluster. When considering the deployment of large SDN, the architecture of multiple controllers is a reasonable choice.

VI. Conclusion

In this paper, we modeled the flow set-up requests from switches to controller as a batch arrival process $M^k/M/1$, then we derived the theoretical expression of average flow service time. Under the circumstance of a limited average flow service time, the number of switches is determined, this provides a method to evaluate the controller capacity. For instance, in a specific context where controller is able to handle 20K new flow requests per second and the average flow arrival rate is 100 per second of each switch, we can obtain that the controller is able to manage 150 access switches if the average flow service time is restricted to 15ms. Particularly, we discussed two issues in determining the flow service time, the two issues are QoS demands of different traffic models and switch buffer size. In addition, we addressed the scene of multiple controllers in SDN. All of these results are meaningful towards the OpenFlow network deployment in the future. However, some problems still exist. For example, the controller placement problem [19] is not considered. This should be a research point in future.

Acknowledgment

This work is supported by Huawei MBB Innovation Research Program.

References

[1] N. McKeown, "Keynote talk: Software-defined networking". In *Proc. IEEE INFOCOM'09* , Rio de Janeiro, Brazil, 19-25 Apr. 2009.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no.2, pp. 69-74, Mar. 2008.

[3] B. Pfaff, "OpenFlow Switch Specification", Version 1.4.0, Oct. 2013. [Online]. Available: http://www.opennetworking.org

[4] S. H. Yeganeh, A. Tootoonchian, Y. Ganjali, "On scalability of software-defined networking". *IEEE Commun. Magazine*, vol. 51, no. 2, pp. 136-141, 2013,

[5] K. Nguyen K, Q. T. Minh, S. Yamada, "A Software-Defined Networking approach for Disaster-Resilient WANs" in *Proc. ICCCN'13*, Nassau, Bahamas, 30 Jul. - 2 Aug. 2013, pp. 1-5.

[6] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Holzle, S. Stuart, adn A. Vahdat, "B4: Experience with a Globally-Deployed Softeware Defined WAN," in *Proc. ACM SIGCOMM conf.*, Hong Kong, China, 12-16 Aug. 2013, pp. 3-14.

[7] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "Openflow switching: Data plane performance," in *Proc. IEEE ICC'10*, Cape Town, South Africa, 23-27 May. 2010, pp. 1-5.

[8] M. Fernandez, "Evaluating OpenFlow Controller Paradigms,". in *Proc. ICN'13*, Seville, Spain, Jan. 27-Feb. 1 2013, pp. 151-157.

[9] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks", in *Proc. Hot-ICE.*, USENIX Association Berkeley, CA, USA, 2012, pp. 10-10.

[10] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an OpenFlow architecture," in *Proc. ITC'11*, San Francisco, CA, USA, 6-8 Sep. 2011. pp. 1-7.

[11] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, "Dynamic Controller Provisioning in Software Defined Networks," in *Proc. IEEE CNSM'13*, University of Zrich, Switzerland, 14-18 Oct. 2013, pp. 18-25.

[12] D. P. Bertsekas, R. G. Gallager, and P. Humblet, "Data networks", 2rd ed. 1992.

[13] D. Manfield and P. Tran-Gia, "Analysis of a Finite Storage System with Batch Input Arising out of Message Packetization," *IEEE Trans. Commun.*, vol 30, no 3, pp. 456-463, Mar. 1982.

[14] ITU-T, "G.1010: End-user multimedia QoS categories," 2001.

[15] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker, "Applying NOX to the Datacenter", in *Proc. HotNets'09*, NY, USA, 22-23 Oct. 2009.

[16] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements and analysis", in *Proc. IMC'09*, Chicago, USA, 4-6 Nov. 2009, pp. 202-208.

[17] A. Lakshmikantha, C. Beck, R. Srikant, "Impact of file arrivals and departures on buffer sizing in core routers", *IEEE/ACM Trans. Net.*, vol 19, no 2, pp. 347-358, Apr. 2011.

[18] "Network Bandwidth Requirements for Media Traffic". [Online]. Available at: http://technet.microsoft.com/en-us/library/jj688118.aspx

[19] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proc. ACM SIGCOMM HoTSDN'12*, Helsinki, Finland, 13-17 Aug. 2012, pp. 7-12.