Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 7659085, 11 pages
https://doi.org/10.1155/2018/7659085

Research Article

Packet Scheduling for Multiple-Switch Software-Defined Networking in Edge Computing Environment

Hai Xue, Kyung Tae Kim, and Hee Yong Youn

College of Information and Communication Engineering, Sungkyunkwan University, Suwon, Republic of Korea

Correspondence should be addressed to Hee Yong Youn; youn7147@skku.edu

Received 23 August 2018; Revised 10 October 2018; Accepted 31 October 2018; Published 18 November 2018

Guest Editor: Jorge Navarro-Ortiz

Software-defined networking (SDN) decouples the control plane and data forwarding plane to overcome the limitations of traditional networking infrastructure. Among several communication protocols employed for SDN, OpenFlow is most widely used for the communication between the controller and switch. In this paper two packet scheduling schemes, FCFS-Pushout (FCFS-PO) and FCFS-Pushout-Priority (FCFS-PO-P), are proposed to effectively handle the overload issue of multiple-switch SDN targeting the edge computing environment. Analytical models on their operations are developed, and extensive experiment based on a testbed is carried out to evaluate the schemes. They reveal that both of them are better than the typical FCFS-Block (FCFS-BL) scheduling algorithm in terms of packet wait time. Furthermore, FCFS-PO-P is found to be more effective than FCFS-PO in the edge computing environment.

1. Introduction

Nowadays, the traditional IP network can hardly deal with the huge volume of traffic generated in the rapidly growing Internet of Things (IoT). As an efficient solution to this issue, a new type of networking paradigm called software-defined networking (SDN) had been proposed [1]. In SDN the control plane and data plane of the network are decoupled, unlike the traditional network. The role of the switches in SDN is delivering the data packets, while the controller is installed separately for controlling the whole network. OpenFlow [2] is one of the most popular protocols developed for the communication between the controller and switch in SDN, and it is only an open protocol [3].

The proliferation of IoT and the success of rich cloud service induced the horizon of a new computing paradigm called edge computing [4]. Here how to efficiently deliver the data from the IoT nodes to the relevant switches is a key issue. While there exist numerous nodes in the edge computing environment, some of them are deployed to detect critical events such as fire or earthquakes. For such peculiar edge nodes, the data must be delivered with the highest priority due to its importance. Therefore, effective priority-based scheduling and a robust queueing mechanism are imperative to maximize the performance of the network where the data processing occurs at the edge of the network. A variety of network traffics are also needed to be considered for accurately estimating the performance [5], allowing early identification of a potential traffic hotspot or bottleneck. This is a fundamental issue in the deployment of SDN for edge computing.

Recently, numerous studies have been conducted to maximize the performance of the controller and OpenFlow switch of SDN. However, to the best of our knowledge, little explorations exist for the performance of SDN switches with priority scheduling. This is an undoubtedly important issue for edge computing since the edge nodes deal with the received data based on the priority. Furthermore, priority scheduling is necessary with multiple switches. In this paper, thus, the priority scheduling scheme for multiple-switch SDN is proposed. Here packet-in messages might be sent to the controller from each of the switches. On the basis of the $M/G/1$ queueing model for the OpenFlow switch, the First Come First Served-Pushout (FCFS-PO) and First Come First Served-Pushout-Priority (FCFS-PO-P) mechanism are proposed for efficiently handling the overload issue. With FCFS-PO, the packets are served based on the order of arrival, while the oldest one is pushed out when the buffer is full.

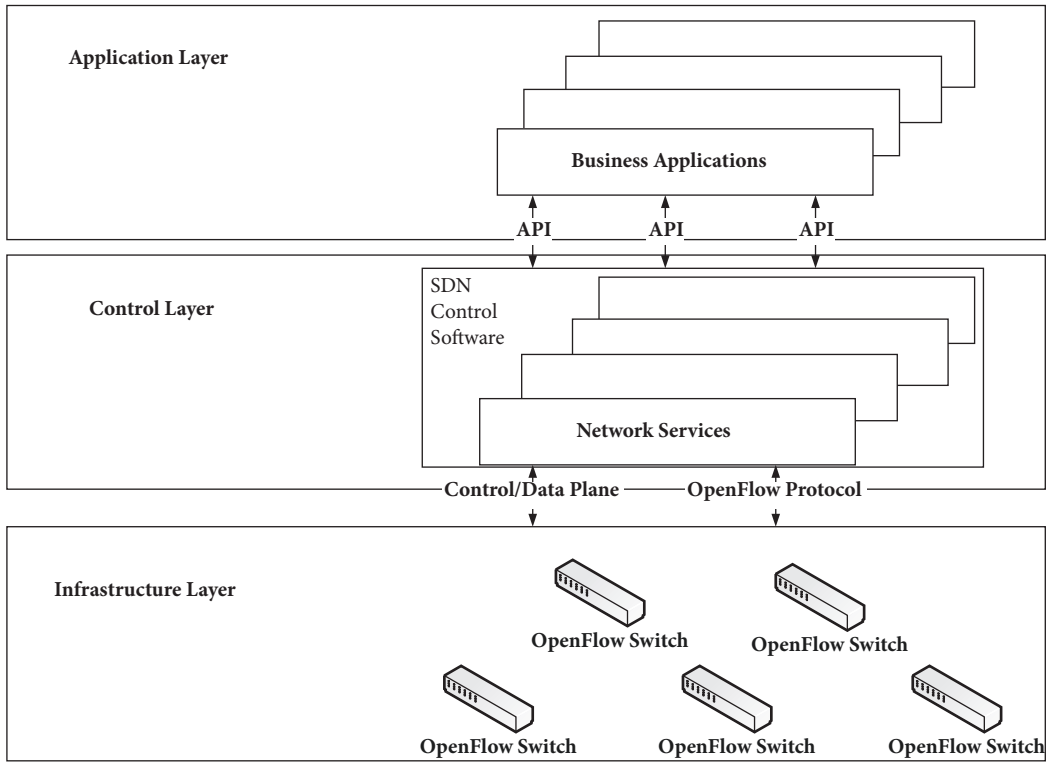

Figure 1: The three-layer structure of SDN.

With FCFS-PO-P, the packets are serviced by the order of the priority on top of the FCFS-PO. Extensive experiment on a testbed reveals that FCFS-PO-P scheduling allows better performance than FCFS-PO in terms of sojourn and wait time for various traffic conditions. Furthermore, both of them display much smaller wait time than the typical FCFS-Block (FCFS-BL) scheduling. The main contributions of the paper are summarized as follows:

(i) The packet scheduling issue is identified with the SDN controller in edge computing environment. The FCFS-PO and FCFS-PO-P scheduling scheme are proposed for coping with this issue

(ii) Analytical model of the proposed scheduling scheme is developed using queueing theory, which can be applied to the evaluation of priority-based scheduling scheme

(iii) The scheduling for solving the packet scheduling problem of SDN controller in edge computing environment is revealed by comparing the proposed method with existing methods

The rest of the paper is structured as follows. Section 2 provides an overview of SDN and edge computing. In Section 3 the priority-based scheduling schemes for multiple-switch SDN are proposed along with their analytical modeling. Section 4 evaluates the performance of the proposed schemes. At last, Section 5 concludes the paper and outlines the future research direction.

2. Related Work

SDN decouples the control logic from the underlying routers and switches. It promotes the logical centralization of network control and introduces the capability of programming the network [6–8]. As a result, flexible, dynamic, and programmable functionality of network operations could be offered. However, the merits are achieved with the sacrifice on essential network performance such as packet processing speed and throughput [9], which is attributed to the involvement of a remote controller for the administration of all forwarding devices.

2.1. SDN. SDN was originated from a project of UC Berkeley and Stanford University [10]. In SDN the network control plane making decisions on traffic routing and load balancing is decoupled from the data plane forwarding the traffic to the destination. The network is directly programmable, and the infrastructure is allowed to be abstracted for flexibly supporting various applications and services. The experts and vendors claim that this greatly simplifies the networking task [11]. There exist three layers in SDN unlike the traditional network of two layers, and a typical structure of SDN based on the OpenFlow protocol is depicted in Figure 1.

The separation of control plane and data plane can be realized by means of a well-defined programming interface between the controller and switches. The controller exercises direct control covering the state of the data plane elements via a well-defined application programming interface (API) as shown in Figure 1. The most notable example of such API is

Table 1: The fields of an entry of a flow table.

Field	Description
Match	Port, packet header, and metadata forwarded from the previous flow table
Priority	Matching precedence of the entry
Counter	Statistics for matching the packets
Instruction	Action or pipeline processing
Timeout	Maximum effective time or free time before the entry is overdue
Cookie	Opaque data sent by the OpenFlow controller

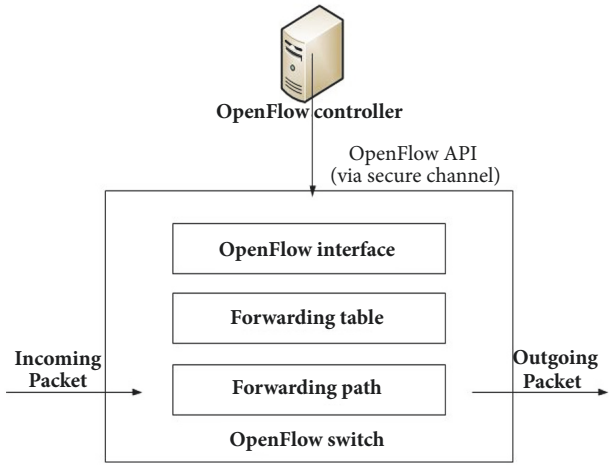

Figure 2: The structure of the OpenFlow protocol.

OpenFlow. There exists one or more tables of packet-handling rules (flow table) in an OpenFlow switch. Each entry of a flow table consists of mainly six fields as listed in Table 1. The rule is matched with a subset of the traffic, and proper actions such as dropping, forwarding, and modifying are applied to the traffic. The flow tables are used to determine how to deal with the incoming packets.

The communication protocol between the controller and switches is particularly important due to the decoupling of the two planes. The OpenFlow protocol is widely used for SDN, which defines the API for the communication between them as Figure 2 illustrates.

In SDN the controller manipulates the flow tables of the switch by adding, updating, and deleting the flow entries. The operation occurs either reactively as the controller receives a packet from the switch or proactively according to the implementation of the OpenFlow controller. A secure channel is supported for the communication between the controller and switch. The OpenFlow switch supports the flow-based forwarding by keeping one or more flow tables [12, 13].

In Xiong et al. [14] a queueing model was proposed for estimating the performance of the SDN controller with the input of a hybrid Poisson stream of packet-in messages. They also modeled the packet forwarding of OpenFlow switches in terms of packet sojourn time [15]. In Sood et al. [16] an analytical model for an SDN switch was developed, which includes the key factors such as flow-table size, packet arrival rate, number of rules, and position of rules. Ma et al. [17] focused on delay estimation with real traffic and end-to-end delay control. A model was proposed in Mahmood et al. [18] which approximates multiple nodes of the data plane as an open Jackson network with the controller. A hybrid routing forwarding scheme as well as a congestion control algorithm was proposed in Shen et al. [19] to solve the traffic scheduling and load balancing problem in software-defined mobile wireless networks. In Miao et al. [20] the performance of SDN was investigated using the Markov Modulated Poisson Process (MMPP) in the presence of bursty and correlated arrivals.

2.2. Edge Computing. With the new era of computing paradigm called edge computing, data are transferred to the network edge for the service including analytics. As a result, computing occurs close to the data sources [21]. The number of sensor nodes deployed on the surroundings is rapidly increasing due to the prevalence of IoT in recent years. Edge computing is an emerging ecosystem which aims at converging telecommunication and IT services, providing a cloud computing platform at the edge of the whole network. It offers storage and computational resources at the edge, reducing the latency and increasing resource utilization. A simplified structure of edge computing is depicted in Figure 3.

As the futuristic vision of IoT, the latest development of SDN could be integrated with edge computing to provide more efficient service. Here how to make the edge nodes more efficient in processing the data is a matter of great concern. Besides, it is essential to accurately estimate the performance of the OpenFlow switch for better usage. While more than thousands of edge nodes may exist in the real environment, some of them are deployed in the critical area whose data must be processed with high priority. Therefore, priority-based scheduling is inevitable to dynamically control the operation of the network considering the property of the traffics.

3. The Proposed Scheme

In this section the proposed scheduling schemes for SDN having multiple switches and the analytical models of the performance are presented. First, the priority scheduling scheme with a single switch and controller is presented. Then, the FCFS-PO and FCFS-PO-P scheme based on multiple switches are introduced which can effectively handle the switch overload issue.

3.1. Priority Scheduling with Single Switch

3.1.1. Basic Structure. SDN allows easy implementation of a new scheme on the existing network infrastructure by decoupling the control plane from the data plane and connecting them via an open interface. The SDN of a single switch and controller is depicted in Figure 4.

Both the controller and switch support the OpenFlow protocol for the communication between them. When a

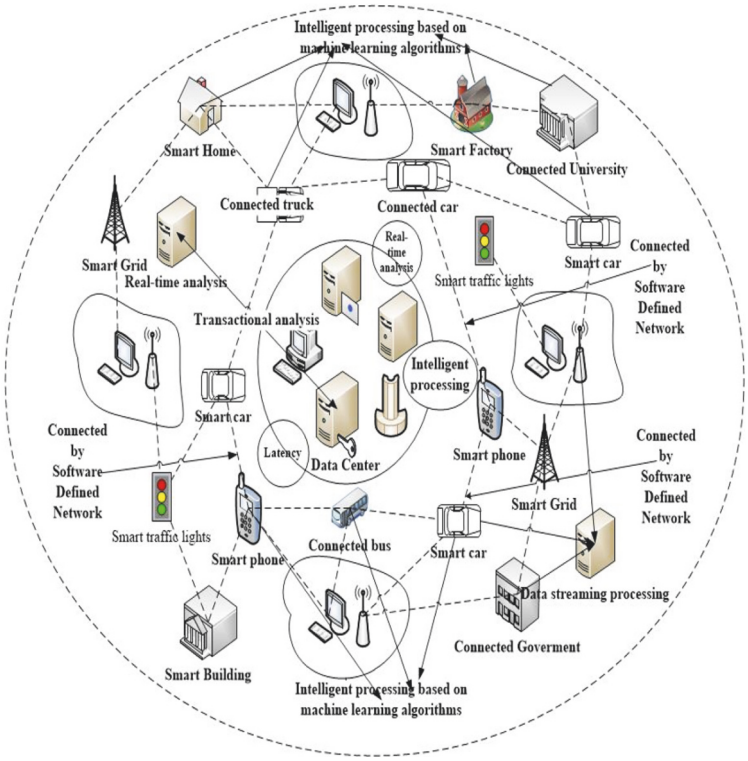

Figure 3: A simplified structure of edge computing.

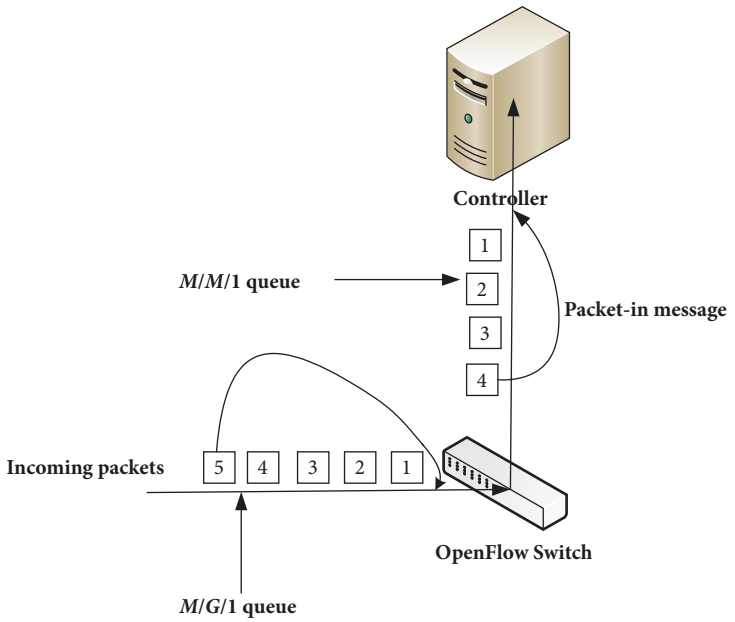

Figure 4: The structure with a single switch and controller.

packet arrives at the OpenFlow switch, the switch performs lookup with its flow tables. If a table entry matches, the switch forwards the packet in the conventional way. Otherwise, the switch requests the controller for the instruction by sending a packet-in message, which encapsulates the packet information. The controller then determines the rule for it, which is installed in other switches. After that, all the packets belonging to the flow are forwarded to the destination without requesting the controller again [15]. The process of packet matching operation is illustrated in Figure 5. Notice from the figure that both the incoming packets and packet-in messages are queued. Here the incoming packets and packet-in

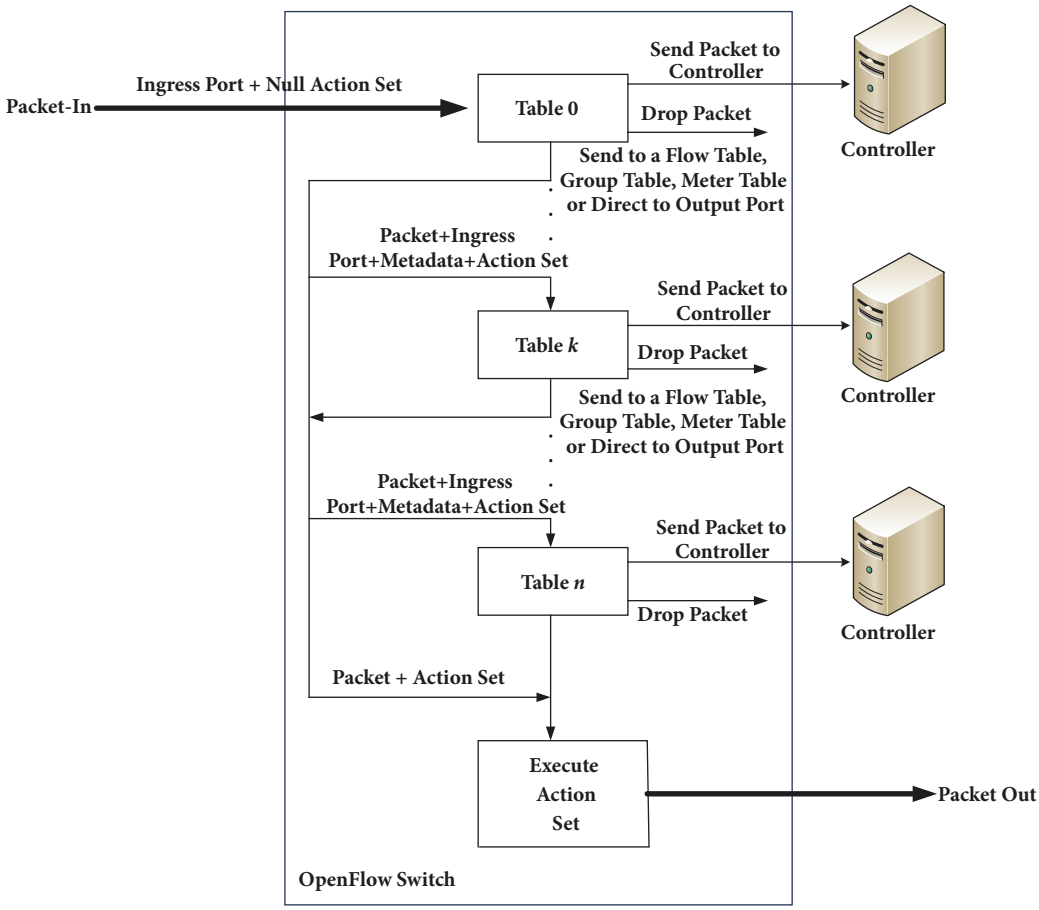

FIGURE 5: The process of packet matching.

messages are handled with the $M/G/1$ model and $M/M/1$ model, respectively, as in the existing researches [14, 16].

3.1.2. Priority-Based Scheduling. Refer to Figure 5. The incoming packets to the OpenFlow switch are queued, and they are scheduled by either the FCFS or FCFS with priority (FCFS-P) scheduling policy. FCFS is applied for regular traffic, while FCFS-P is applied for urgent packets.

In the FCFS $M/G/1$ queueing system, the average sojourn time of a packet consists of two components. The first component is the time required for the processing of the packets which are waiting in the queue at the time of the packet arrival, and the second one is the time due to the packet which is in service. The second component is nonzero if the system is busy, while it is zero if the system is empty. Considering the two components, the average sojourn time of a packet with FCFS scheduling is obtained as follows. With the $M/G/1$ queue, λ is the arrival rate of the Poisson process and $\overline{x}$ is average service time. The models in this subsection can be referred to in [22].

$$S = N_q \overline{x} + \rho \frac{\overline{x}^2}{2\overline{x}}. \tag{1}$$

Here N_q is the number of packets in the queue, ρ is the probability that the queue is busy, and $x^2/(2\overline{x})$ is the average residual lifetime of the service. By adopting Little's equation and the P-K (Pollaczek and Khinchin) equation, (1) can be expressed as follows:

$$S = x + \frac{\lambda \overline{x}^2}{2(1-\rho)}. \tag{2}$$

The model for FCFS-P is different from the FCFS $M/G/1$ queue due to the out-of-order operation. There exist several classes of packets having different Poisson arrival rates and service times. Assuming that there are m classes, $\overline{x}_i$ and λ_i are average service time and arrival rate of class-i, respectively. The time fraction of the system working on class-i is then $\rho_i = \lambda_i \overline{x}_i$. The packets are processed according to the priority with preemptive node, while the class of a smaller number is given higher priority.

In the $M/G/1$ queueing system of FCFS-P, the average sojourn time of an incoming packet consists of three components. The first component is the time taken for the packet currently in service to be finished, S_0. The second one is the time taken for the service of the packets waiting in the queue whose priority is higher than or equal to it. The third component is for the service of the packets arriving after itself but having higher priority. The number of packets of class-m is $\lambda_m S_m$ according to Little's result theorem [23]. Hence,

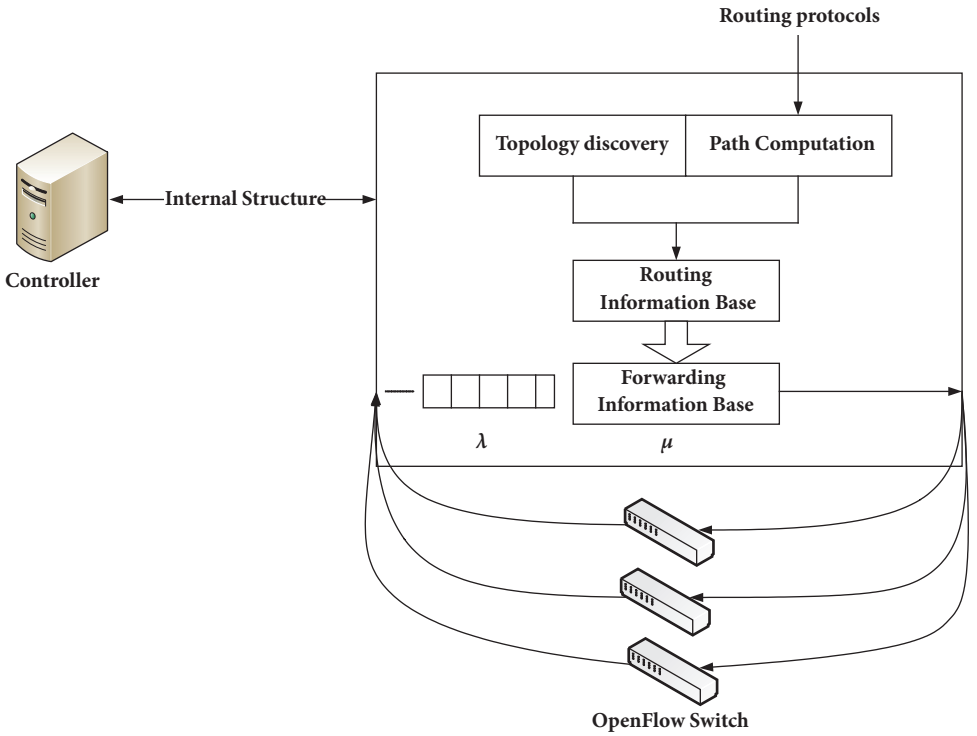

Figure 6: The processing of packet-in messages in the SDN controller.

the sojourn time of the FCFS-P $M/G/1$ queueing system is as follows:

$$S_m = S_0 + \sum_{i=1}^{m} \overline{x}_i \left(\lambda_i S_i\right) + \sum_{i=1}^{m-1} \overline{x}_i \left(\lambda_i S_i\right). \tag{3}$$

Equation (3) can be solved by recursively applying the equation as follows:

$$\sigma_m = \sum_{i=1}^{m} \rho_i. \tag{4}$$

$$S_m = \frac{S_0}{\left(1 - \sigma_m\right)\left(1 - \sigma_{m-1}\right)} \tag{5}$$

With preemptive scheduling, S_0 is due to the packets of higher priority. Therefore, it can be expressed as follows:

$$S_0 = \sum_{i=1}^{m} \rho_i \frac{\overline{x}_i^2}{2\overline{x}_i}. \tag{6}$$

3.1.3. Packet-In Messages. As previously stated, the OpenFlow switch sends a packet-in message to the relevant controller as a flow setup request when a packet fails to match the flow table. The diagram of the processing of the packet-in message is illustrated in Figure 6. Notice that the process of packet-in message has a one-to-one correspondence with the process of flow arrival regardless of the number of switches connected to the controller. Each flow arrival is assumed to follow the Poisson distribution. Therefore, the packet-in messages coming from several switches constitute a Poisson stream by the superposition theorem [15].

3.2. Priority Scheduling with Multiple Switches

3.2.1. Basic Structure. Since SDN decouples the control plane and data forwarding plane, the data transmission consists of two types. One is from the edge nodes to the switches, and the other one is from a switch to the remote controller for the packet-in message. They are illustrated in Figure 7.

There is at least one flow table in an OpenFlow switch, and the incoming packets are matched from the first flow table. Upon a match with a flow entry, the instruction set of the flow entry is executed. On table miss, there are three options: dropping, forwarding to the subsequent flow tables, and transmitting to the controller as a packet-in message. One option is selected by the programmer when the flow tables are sent to the OpenFlow switches.

3.2.2. FCFS-PO Scheduling. With FCFS-PO, if the buffer is full when a packet enters, it is put at the tail of the queue while the packet at the head is pushed out. All the packets move forward one position when a packet is served. Here the position of a packet in the buffer is decided by the number of waiting packets.

The FCFS-PO mechanism is modeled assuming that the packets arriving at the switch follow the Poisson arrival with the rate of $\lambda(\lambda > 0)$. This means that the interarrival times are independent and follow exponential distribution. The packet service times are also independent and follow general

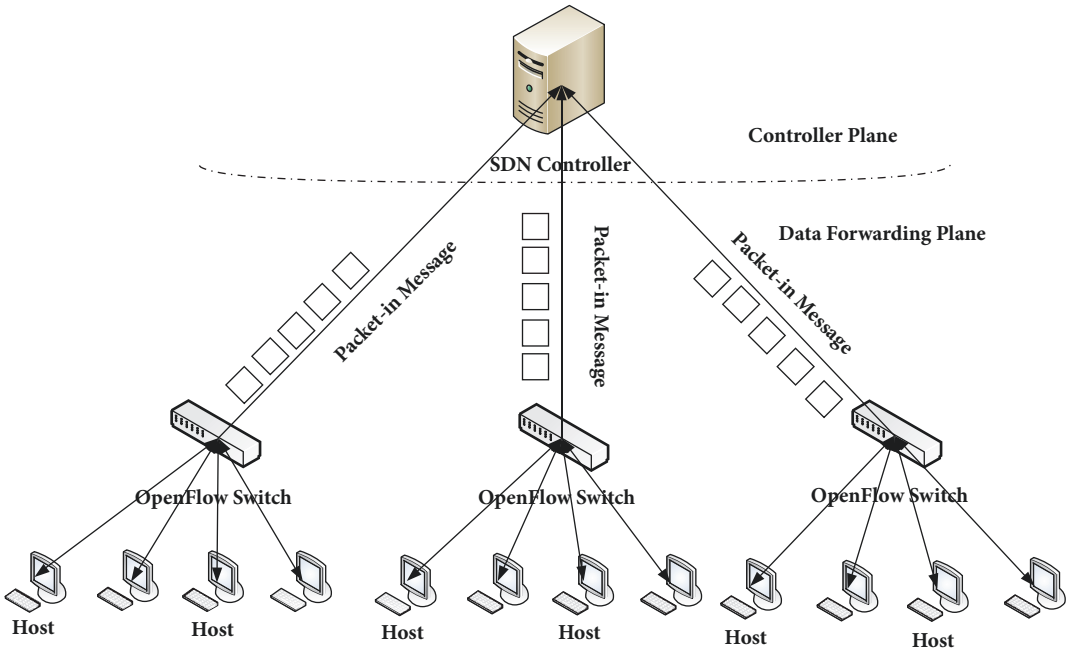

Figure 7: The structure of multiple-switch SDN.

distribution $(1/\mu)$. Also, assume that the system can contain N packets at most and the buffer size is $(N-1)$. At last, the arrival process and service process are assumed to be independent.

Assume that $Q_i(1 \le i \le (N-1))$ is the steady state probability of the $M/G/1/N$ queue, while $P_i(1 \le i \le (N-1))$ represents the steady state probability of the $M/G/1$ queue. Q_i is obtained as follows using the approach employed in [24]:

$$Q_i = \frac{P_i}{\sum_{i=0}^{N-1} P_i}, \quad 1 \le i \le (N-1). \tag{7}$$

$Q_a(1 \le a \le N)$ is the steady state probability of a packets already in the system when a new packet arrives.

Then, according to the Poisson Arrivals See Time Averages (PASTA) character [24], the following equation can be obtained:

$$\begin{aligned} Q_a &= \frac{Q_a}{Q_0 + \rho}, \quad a = 0, 1, 2, \ldots, (N-1) \\ Q_N &= 1 - \frac{1}{Q_0 + \rho}, \quad a = N. \end{aligned} \tag{8}$$

Here $\rho = (\lambda/\mu)$. And two performance indicators of the queue in the steady state are dealt with as follows:

(i) The probability of the packets to get the service, P_x,

$$P_x = \frac{1}{(Q_0 + \rho)} \tag{9}$$

(ii) The packet loss rate of the system, P_l,

$$P_l = 1 - P_x = 1 - \frac{1}{Q_0 + \rho} \tag{10}$$

With the FCFS-PO mechanism, when the buffer is full of packets, the incoming packet will be put at the end, and the buffer management policy will push out the head-of-line (HOL) packet for the newly incoming packet. Let us denote by $P_s(a,b)$ the steady state probability for the packet in state (a,b), and it finally gets the service when a packet leaves the system. When $a = 1$, the packet is in service. When $2 \le a \le N$ and $0 \le b \le (N-a)$, $p_s(a,b)$ needs to wait for the service. Assume that there are m incoming packets while a packet is in service. Then, $m \le (N-b-2)$ for $p_s(a,b)$ to get the service. If $m \le (N-a-b)$, there is no packet to be pushed out. Then, the position of the packet is changed from (a,b) to $(a-1, b+m)$. The buffer is full while $(N-a-b)+1 \le m \le (N-b-2)$, and the incoming packet pushes out the packet at position_2. Then, the position of the packet is changed from (a,b) to $(N-m-b-1, b+m)$. Finally, the state-transition equation of $P_s(a,b)$ is obtained as follows [25]:

$$\begin{aligned} &P_s(a,b) \\ &\quad = \prod \alpha_m \bullet (m \mid (a,b)) \\ &\qquad + \sum_{m=N-a-b+1}^{N-b-2} \alpha_m \bullet P_s(N-m-b-1, b+m). \end{aligned} \tag{11}$$

Here α_m can be obtained from the result of imbedded Markov points in [26]. Then, the average waiting time is analyzed. The following Boolean function is used to show if the packet was served or not:

$$L = \begin{cases} 0 & \textit{Served} \\ 1 & \textit{Not served}. \end{cases} \tag{12}$$

Using (9) and (10), the probability of $P(L = 0)$ and $P(L = 1)$ functions can be easily obtained.

Let B_n $(n \geq 1)$ represent the number of incoming packets while n packets are being served, and $R(t)$ is the remaining time of the current packet in service. The Laplace-Stieltjes Transform (LST) of b_a is obtained as follows:

$$b_a(\theta) = \left[\int_0^{\infty} \frac{(\lambda c)^a}{a!} e^{-(\lambda+\theta)c} dR(t) \mid \frac{Q_n + \rho - 1}{Q_n + \rho}\right] \cdot \left(\frac{Q_n + \rho - 1}{Q_n + \rho}\right). \tag{13}$$

Denote by $H(a,b)$ the average waiting time for $P_s(a,b)$ until it finally gets the service, and the waiting time is no longer than t. Then, the probability of $H(a,b,\theta)$ is shown as follows:

$$H(a,b,\theta) = \int_0^{\infty} e^{-\theta t} dH(a,b,\theta). \tag{14}$$

So, $H(a,b)$ can be obtained:

$$H(a,b) = -\lim_{\theta \longrightarrow 0} H(a,b,\theta). \tag{15}$$

The conditional average waiting time for $p_s(a,b)$ is as follows:

$$\begin{aligned} T = \rho \bullet \sum_{a=1}^{N-1} \sum_{b=0}^{N-a-1} Q_a \Big[& b_a \bullet H(a,b) + P_s(a,b) \\ & \bullet \frac{(b+1)}{\lambda} \bullet b_{b+1} \Big] + \rho \bullet \sum_{a=1}^{N-1} \sum_{b=N-a}^{N-2} Q_a \Big[b_a \\ & \bullet H(N-b-1,b) + P_s(N-b-1,b) \\ & \bullet \frac{(b+1)}{\lambda} b_{b+1} \Big] + \rho \bullet \sum_{b=0}^{N-2} Q_N \Big[b_a H(N-b-1,b) \\ & + P_s(N-b-1,b) \bullet \frac{(b+1)}{\lambda} \bullet b_{b+1} \Big]. \end{aligned} \tag{16}$$

Here Q_a and $b_a (1 \leq a \leq N)$ are given by (8) and (13), respectively. $H(a,b)$ is given by (15).

3.2.3. FCFS-PO-P Scheduling. In the modeling of FCFS-PO-P, the arrival process and service process of the incoming packets are the same as the modeling of FCFS-PO. In this paper, assume that the newly incoming packet has the highest priority among the packets already in the queue. Then, the newly incoming packet will be put in the front position to get service first. And the buffer management policy pushes out the packet in position_N which waited longest in the queue. Let $P_s(a)$ denote the steady state probability of packet in position_a and it is finally served. With the FCFS-PO-P policy, $p_s(a) (2 \leq a \leq N)$ waits in the sequence. Assume that m incoming packets arrived while a packet is in service. Then, $m \leq (N - a)$ for the packet to be served, and $p_s(a)$ moves to position_$(a+m-1)$ when the service ends. The state-transition equation of $P_s(a)$ is as follows [25]:

$$\begin{aligned} P_s(a) = & \prod \alpha_m \bullet (m \mid (a,b)) \\ & + \sum_{m=0}^{N-a} \alpha_m \bullet P_s(a+m-1), \quad a = 2,3,\ldots,N. \end{aligned} \tag{17}$$

Similarly, α_m can be obtained from [26]. Now, the average waiting time is analyzed. The probability of $p_s(a)$ to get the service and the service time not being larger than t is $G(a,c)$. Assume that the following equation is the LST of $G(a,c)$:

$$h(a,\theta) = \int_0^{\infty} e^{-\theta c} dG(a,c). \tag{18}$$

Similar to (17), the recursive equation of $h(a,\theta)$ is obtained:

$$h(a,\theta) = \sum_{m=0}^{N-a} \alpha_m(\theta) \bullet h(a+m-1,\theta), \quad a = 2,3,\ldots,N. \tag{19}$$

Let $H(a)$ denote average waiting time for $p_s(a)$ until it finally gets the service. Similar to (14) and (15), $H(a)$ can be obtained as follows:

$$H(a) = -\lim_{\theta \longrightarrow 0} h(a,\theta). \tag{20}$$

At last, the average waiting time for $p_s(a)$ is as follows:

$$T = \rho \bullet \sum_{a=0}^{N-2} b_a H(a+1) + P_s(a+1) \bullet \frac{(a+1)}{\lambda} \bullet b_{a+1}. \tag{21}$$

4. Performance Evaluation

In this section the performance of the proposed scheduling schemes for the OpenFlow switch is evaluated.

4.1. Experiment Environment. A testbed is implemented with three Raspberry Pi nodes for the experiment. The data with the priority scheduling are collected from the testbed, and they are compared with the analytical models. Open vSwitch is installed in the Raspberry Pi node to implement the real switch with the OpenFlow protocol. Open vSwitch is a multilayer, open source virtual switch targeting all major hypervisor platforms. It was designed for networking in virtual environment, and thus it is quite different from the traditional software switch architecture [27]. The parameters of the Raspberry Pi nodes are listed in Table 2.

For the experiment one remote controller and three OpenFlow switches are implemented. Three Raspberry Pi nodes operate as the OpenFlow switches, and Floodlight was installed in another computer working as a remote controller. Floodlight is a Java-based open source controller, and it is one of the most popular SDN controllers supporting physical and virtual OpenFlow compatible switches. It is based on the Bacon controller from Stanford University [28]. Note that

Table 2: The parameters of Raspberry Pi node.

Version	Raspberry Pi 3 Model B
SoC	BCM2837
CPU	Quad Cortex A53@1.2GHZ
Instruction set	ARMv8-A
GPU	400MHz VideoCore IV
RAM	1GB SDRAM
Storage	32G
Ethernet	10/100

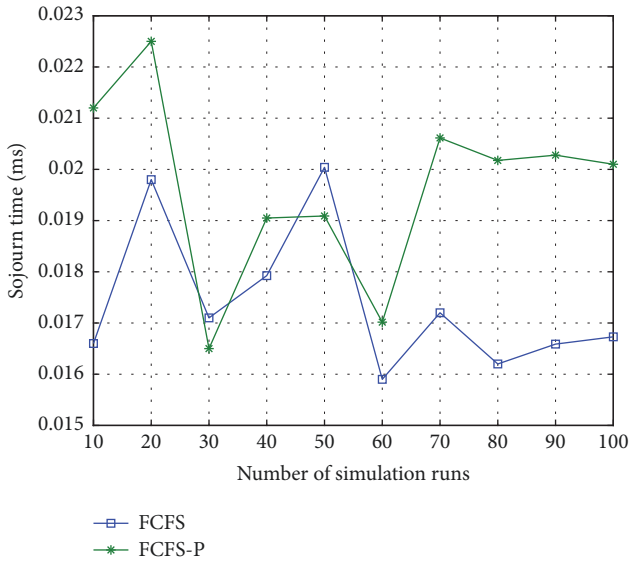

Figure 8: The comparison of sojourn times with FCFS and FCFS-P.

four network interfaces are supplied with the Raspberry Pi node. Therefore, three OpenFlow switches are installed for collecting data, and one interface is used to be connected to the controller network. The network traffics are generated by the software tool "iPerf" [29]. One host is selected as the server and another as the terminal by iPerf. Then, the host sends packets to the server by the UDP protocol in the bandwidth of 200Mbps.

4.2. Simulation Results. First, the simulation results of the proposed scheduling algorithm with a single switch are presented. Figure 8 compares the FCFS and FCFS-P scheduling algorithm. Notice that the FCFS-P algorithm mostly causes longer sojourn time than the FCFS algorithm. The FCFS-P algorithm occasionally displays similar performance to FCFS when the incoming packets have high priority.

Figure 9 compares the data from the experiment and analytical model. As aforementioned, three variables, λ, $\overline{x}$, and ρ, are used in the FCFS scheduling algorithm. Similar to the previous research [13], λ and $\overline{x}$ are empirically set to be 2 and 0.016, respectively. ρ is set to be 0.1, 0.5, and 0.9 to check the performance with various conditions. It reveals that

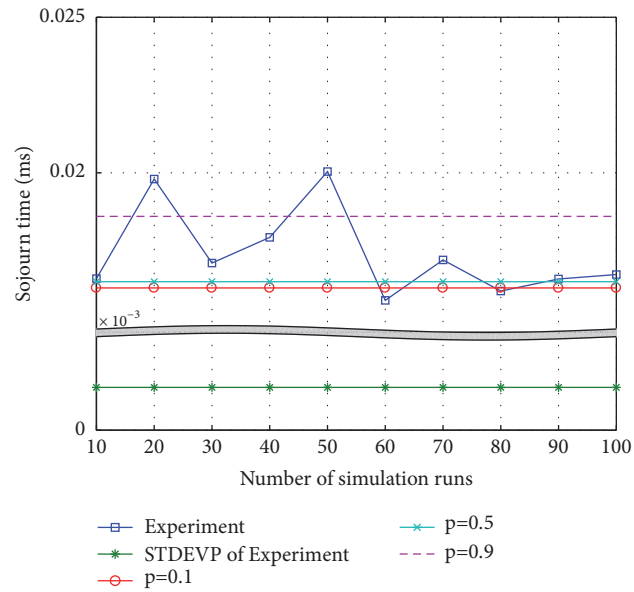

Figure 9: The comparison of the data from the experiment and analytical model with FCFS.

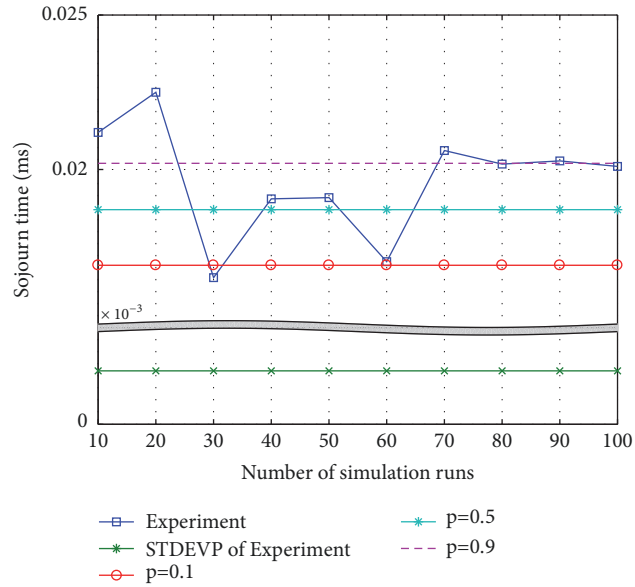

Figure 10: The comparison of the data from the experiment and analytical model with FCFS-P.

the data stabilizes as the simulation time passes, and ρ of 0.5 allows the closest estimation.

Figure 10 is for the case of FCFS-P. Here the values of ω_0, $\overline{x}_i$, λ_i, and S_i are 0.02, 0.018, 0.016, and 2, respectively. In this case, ρ of 0.9 shows the best result.

Assume that $N = 10$ and $\mu = 5$. Figure 11 compares FCFS-PO and FCFS-PO-P as λ varies. Observe from the figure that the average waiting time with FCFS-PO-P is smaller than with the FCFS-PO mechanism.

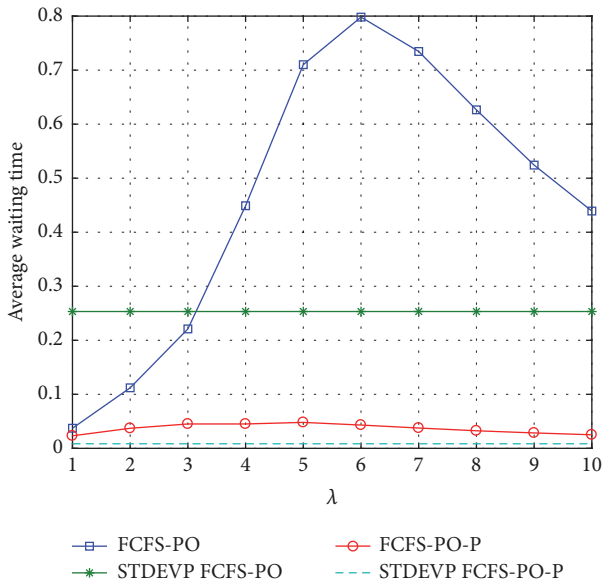

Figure 11: The comparison of waiting times with FCFS-PO and FCFS-PO-P.

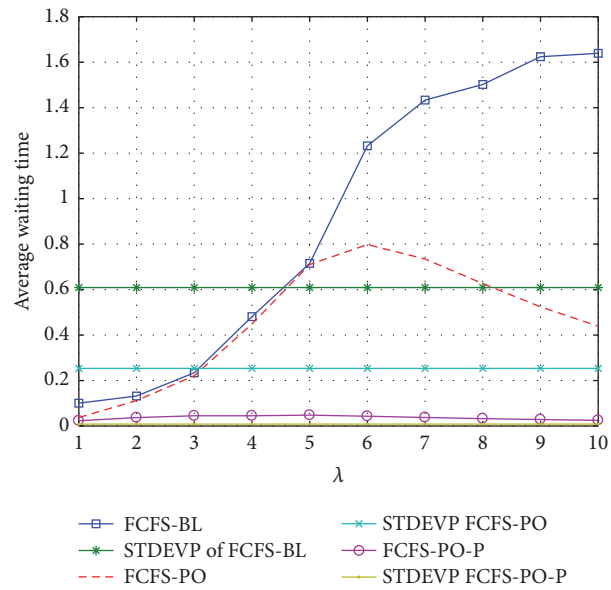

Figure 12: The comparison of the waiting time of the three mechanisms.

The probability density function of wait time with FCFS-PO in the steady state is as follows [30]:

$$W(t) = \sum_{n=1}^{N-1} V_n \left[B^{(n-1)}(t) \bullet R(t) \right] + V_0. \quad (22)$$

Here $B^{(n)}(t)$ is the n-fold convolution of $B(t)$ and $B(t) \cdot R(t)$ is the convolution of $B(t)$ and $R(t)$. The incoming packet will be processed directly if there is no packet waiting in the system, by which means the waiting time of the packet is 0. V_0 is the probability of the system to be idle and V_n is the probability that there are $n(1 \le n \le (N-1))$ packets waiting in the system. Then, the incoming packet will wait at position_$(n+1)$. The total wait time consists of the remaining service time and the service time of the packet in front of it. The average wait time can then be obtained as follows:

$$T(t) = \int_0^{\infty} t dW(t). \quad (23)$$

The FCFS-PO-P mechanism is compared with FCFS-BL and FCFS-PO in Figure 12. With FCFS-BL, the incoming packet is lost if the buffer is full. Notice from the figure that the pushout mechanism is more effective than the BL mechanism. Particularly, the superiority gets higher as λ increases. Generally speaking, the FCFS-PO-P mechanism is the best among the three mechanisms.

5. Conclusion

In this paper the FCFS-PO and FCFS-PO-P scheduling algorithms for multiple-switch SDN have been proposed targeting the edge computing environment. Since there exist some nodes requiring urgent processing in this environment, the priority-based scheduling approach was proposed. Here some switches might be overloaded if the packet arrival rate is high, and the proposed FCFS-PO and FCFS-PO-P mechanism are able to effectively deal with the situation. The SDN environment was implemented with three Raspberry Pi node switches and one independent computer of remote controller, and the sojourn time and wait time were analyzed. The measured data from the testbed were compared with those from the analytical models to validate them. The experiment results show that the FCFS-PO-P scheduling is better than the FCFS-PO scheduling based on the wait time. The comparison with the existing FCFS-BL scheduling algorithm reveals that FCFS-PO-P scheduling is the best among them, while FCFS-BL is the worst.

Several parameter values have been set empirically for the network operation. In the future the analytical models will be developed by which the value of the parameters can be properly decided. Also, for a more comprehensive understanding of the scheduling issues in SDN, the process of packet-in messages based on the $M/M/1$ model will be developed.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partly supported by the Institute for Information & Communications Technology Promotion (IITP)

grant funded by the Korea government (MSIT) (No. 2016-0-00133, research on edge computing via collective intelligence of hyperconnection IoT nodes), Korea, under the National Program for Excellence in SW supervised by the IITP (Institute for Information & Communications Technology Promotion) (2015-0-00914), the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2016R1A6A3A11931385, research of key technologies based on software-defined wireless sensor network for real-time public safety service; 2017R1A2B2009095, research on SDN-based WSN supporting real-time stream data processing and multiconnectivity), the second Brain Korea 21 PLUS project, and Samsung Electronics.

References

[1] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[2] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow: enabling innovation in campus networks," *Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[3] Open Networking Foundation". Available at http://www.opennetworking.org, access March 2014.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[5] J. Ansell, W. Seah, B. Ng, and S. Marshall, "Making Queueing Theory More Palatable to SDN/OpenFlow-based Network Practitioners," in *Proceedings of the 8th Internatioanl Workshop on Management of the Future Internet (ManFI)*, pp. 1119–1124, 2016.

[6] H. Farhady, H. Lee, and A. Nakao, "Software-Defined Networking: a survey," *Computer Networks*, vol. 81, pp. 79–95, 2015.

[7] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[8] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: state of the art and research challenges," *Computer Networks*, vol. 72, pp. 74–98, 2014.

[9] A. Gelberger, N. Yemini, and R. Giladi, "Performance analysis of Software-Defined Networking (SDN)," in *Proceedings of the 2013 IEEE 21st International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication, MASCOTS 2013*, pp. 389–393, USA, August 2013.

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

[11] M.-K. Shin, K.-H. Nam, and H.-J. Kim, "Software-defined networking (SDN): A reference architecture and open APIs," in *Proceedings of the 2012 International Conference on ICT Convergence: "Global Open Innovation Summit for Smart ICT Convergence", ICTC 2012*, pp. 360-361, October 2012.

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

[13] F. R. B. Cruz and T. Van Woensel, "Finite queueing modeling and optimization: A selected review," *Journal of Applied Mathematics*, vol. 2014, 2014.

[14] K. Choi, Y. Suh, and D. Yoo, "Extended collaborative filtering technique for mitigating the sparsity problem," *International Journal of Computers Communications & Control*, vol. 11, no. 5, p. 631, 2016.

[15] B. Xiong, K. Yang, J. Zhao, W. Li, and K. Li, "Performance evaluation of OpenFlow-based software-defined networks based on queueing model," *Computer Networks*, vol. 102, pp. 172–185, 2016.

[16] K. Sood, S. Yu, and Y. Xiang, "Performance Analysis of Software-Defined Network Switch Using M/Geo/1 Model," *IEEE Communications Letters*, pp. 114–119, 2016.

[17] H. Ma, J. Yan, P. Georgopoulos, and B. Plattner, "Towards SDN based queuing delay estimation," *China Communications*, vol. 13, no. 3, pp. 27–36, 2016.

[18] K. Mahmood, A. Chilwan, O. Østerbø, and M. Jarschel, "Modelling of OpenFlow-based software-defined networks: The multiple node case," *IET Networks*, vol. 4, no. 5, pp. 278–284, 2015.

[19] D. Shen, W. Yan, Y. Peng, Y. Fu, and Q. Deng, "Congestion Control and Traffic Scheduling for Collaborative Crowdsourcing in SDN Enabled Mobile Wireless Networks," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[20] W. Miao, G. Min, Y. Wu, H. Wang, and J. Hu, "Performance Modelling and Analysis of Software Defined Networking under Bursty Multimedia Traffic," *ACM Transactions on Multimedia Computing Communication, and Applications*, vol. 12, no. 5s, pp. 77-19, 2016.

[21] M. Patel and A. Pandya, "Edge Computing: Design a Framework for Monitoring Performance between Datacenters and Devices of Edge Networks," *International Journal of Computer & Mathematical Sciences*, vol. 6, no. 6, pp. 73–77, 2017.

[22] D. Finkel, "Book review: Fundamentals of Performance Modeling by M.K. Molloy (Macmillan, 1989)," *ACM SIGMETRICS Performance Evaluation Review*, vol. 18, no. 3, p. 23, 1990.

[23] J. D. C. Little, "Little's Law as viewed on its 50th anniversary," *Operations Research*, vol. 59, no. 3, pp. 536–549, 2011.

[24] J. Shortle, J. Thompson, D. Gross, and C. Harris, *Fundamentals of Queueing Theory*, John Wiley Sons, New York, USA.

[25] Y. Lee, B. Choi, B. Kim, and D. Sung, "Delay Analysis of an $M/G/1/K$ Priority Queueing System with Push-out Scheme," *Mathematical Problems in Engineering*, 2007.

[26] S. Kasahara, H. Takagi, Y. Takahashi, and T. Hasegawa, "M/G/L/K System with Push-Out Scheme Under Vacation Policy," *Journal of Applied Mathematics and Stochastic Analysis*, vol. 9, no. 2, pp. 143–157, 1996.

[27] B. Pfaff, J. Pettit, T. Koponen et al., "The design and implementation of open vSwitch," in *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2015*, pp. 117–130, USA, May 2015.

[28] H. Akcay and D. Kaplan, "Web-Based User Interface for the Floodlight SDN Controller," *International Journal of Advanced Networking and Applications*, vol. 08, no. 05, pp. 3175–3180, 2017.

[29] H. Zheng, Y. Zhao, X. Lu, and R. Cao, "A Mobile Fog Computing-Assisted DASH QoE Prediction Scheme," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[30] M. Rawat, H. Rawat, and S. Ansari, "Analysis of GI/GI/1 Queue by Push-Out Simulation Technique," *International Journal of Scientific and Research Publications*, vol. 3, no. 6, pp. 1–4, 2013.