Buffer Size Evaluation of OpenFlow Systems in Software-Defined Networks

Ayan Mondal, *Student Member, IEEE*, Sudip Misra, *Senior Member, IEEE*, and Ilora Maity, *Student Member, IEEE*

***Abstract*—In this paper, we address the problem of minimum buffer size evaluation of an OpenFlow system in software-defined networks (SDNs), while ensuring optimum packet waiting time. The problem is important, as OpenFlow is one of the popular southbound application programing interfaces, which enables controller–switch interaction. The related existing literature addresses schemes on enhancement and packet flow in an OpenFlow system. However, there is a need to analyze the optimum buffer size of an OpenFlow switch, for ensuring the quality-of-service of SDNs. In this paper, we propose an analytical scheme for buffer bound evaluation of an OpenFlow system, named OPUS. Additionally, we propose a queuing scheme for an OpenFlow system—C-M/M/1/K/∞ queuing model—based on the OpenFlow specification version 1.5.0. Further, we calculate the minimum buffer size requirement of an OpenFlow switch, theoretically. Simulation-based analysis exhibits that with two times increase in packet processing rate, the packet arrival rate can be increased by 26.15–30.4%. We infer that for an OpenFlow system, the minimum buffer size is 0.75 million packets with the maximum packet arrival and the minimum processing rate of 0.20–0.25 million packets per second (mpps) and 0.03–0.35 mpps, respectively, and the maximum packet waiting time is 0.173–0.249 s.**

***Index Terms*—Analytical evaluation, buffer size, OpenFlow, queuing theory, software-defined network (SDN).**

I. Introduction

SOFTWARE-defined networks (SDNs) decouple the network control, and the packet forwarding and processing tasks [1] into the *control* and the *data planes*. The control plane includes northbound and southbound application programing interfaces (APIs). Presently, OpenFlow is one of the popular southbound APIs for controller–switch interaction in the SDN architecture. In OpenFlow systems, an OpenFlow switch contains one or more flow-tables to store packet forwarding rules. The flow-tables are of two types such as ingress and egress flow-tables. Each flow-table contains a set of flow rules. On the other hand, ingress buffers are associated with a finite buffer to store the incoming packets. From each ingress buffer, the packet gets forwarded to match against the flow-table entries. After finding the match in the ingress flow-table, each packet gets forwarded to the egress flow-table, if the egress flag of the packet is set. Thereafter, the packets get forwarded to the output port.

Manuscript received September 8, 2017; revised February 20, 2018; accepted March 22, 2018. (*Corresponding author: Sudip Mishra.)*

The authors are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, India (e-mail: ayanmondal@iitkgp.ac.in; smisra@sit.iitkgp.ernet.in; imaity@iitkgp.ac.in).

Digital Object Identifier 10.1109/JSYST.2018.2820745

In the existing literature, researchers proposed different schemes and architecture for SDNs, viz., [1]–[4], which are supported by the OpenFlow protocol and switches. The proposed approaches depend on optimum values of buffer size, packet arrival, and processing rates. To the best of our knowledge, in the existing literature, there is no analytical model on the evaluation of the minimum buffer size of an OpenFlow switch in OpenFlow systems. There is a need for an analytical model to evaluate the minimum buffer size requirement of an OpenFlow switch for ensuring quality-of-service with the minimum packet drop in OpenFlow systems. The model is to be used for evaluating the maximum arrival rate, the minimum processing rate, and the minimum buffer size of an OpenFlow switch. Moreover, the analytical model estimates the maximum packet waiting time in an OpenFlow switch. In this work, we model packet flow through an OpenFlow switch as a *Markovian* process. Hence, we consider that the packet arrival to an OpenFlow switch as a *Poisson* arrival process. Additionally, we consider that the service time of each packet at the OpenFlow switch follows an *exponential* distribution [5]. To the best of our knowledge, this is the first queuing theory-based model for evaluating the optimum buffer size of an OpenFlow switch-based system with multiple ingress buffers.

We evaluate the optimum values of different performance metrics such as buffer size, packet arrival and processing rates, and packet waiting time, in case of packet flow through an OpenFlow switch-based system. The primary *contributions* of our work are summarized below.

1) Initially, a queuing theory-based analytical scheme, named *OPUS*, based on the existing OpenFlow protocol [6] is developed. This model depicts events such as the packet arriving at an ingress port, packets getting queued at ingress buffers, and packets getting processed by an OpenFlow switch in OpenFlow systems.
2) We perform a queuing theory analysis of the proposed model, OPUS. Based on the analysis, we comment on different events such as the optimum buffer size, traffic intensity, and the packet waiting time.
3) Finally, in a simulated environment, we estimated the optimal value of buffer size, and the packet arrival and processing rates of an OpenFlow switch-based system. Additionally, we evaluated the maximum packet waiting time of an OpenFlow switch in OpenFlow systems.

II. Related Works

This section gives an overview of the related work. The existing literature are discussed in different categories—

1) enhancement of OpenFlow-enabled networks, 2) performance analysis of OpenFlow-enabled networks, and 3) analysis of buffer behavior in other domains of networking.

Meiners *et al.* [4] proposed a technique to compress flow-table entries and increase storage space in an OpenFlow switch. Congdon *et al.* [2] presented a per-port optimization method to reduce switch latency and power consumption. In another work, Mogul *et al.* [7] suggested a hashing-based method to reduce flow-table lookups in an OpenFlow switch. Reitblatt *et al.* [8] addressed the issues of consistent network updates. The authors proposed a set of abstract operations to change the network configuration such that each incoming packet follows either the old configuration or the new one. Wang *et al.* [9] proposed a network storage approach without any physical storage with the help of SDN. Li *et al.* [10] reduced the communication overhead of SDN, while using buffer in an SDN switch, and proposed to store the packet header instead of the entire packet. Bera *et al.* [11] proposed an SDN-based wireless sensor network for provisioning application-aware service in Internet of Things. Hayes *et al.* [12] studied the traffic-classification in SDN. Katta *et al.* [3] addressed the tradeoff between network update duration and rule-space overhead.

Although there exist prior works dealing with different aspects of SDN and OpenFlow, very few works focus on performance analysis of OpenFlow enabled networks. Jarschel *et al.* [13] modeled the OpenFlow architecture as an M/M/1 forward queuing system and an M/M/1-S feedback queuing system. This model analyzes the probability of packet drop and estimates the total sojourn time of a packet as well as the delay at an OpenFlow switch-based system, while assuming an infinite buffer size per switch. As this work is based on OpenFlow protocol version 1.0.0, the authors considered that there exists a single flow-table per OpenFlow switch. However, according to the OpenFlow version 1.5.0 [6], each switch has one or more flow-tables. Metter *et al.* [14] proposed an M/M/∞ queuing model-based analytical model to analyze a tradeoff between signaling rate and switch table occupancy and calculate an optimum flow-rule time-out period. In another work, Azodolmolky *et al.* [15] proposed a network calculus-based model for SDN. This model depicts controller–switch interaction and performs an analysis of network performance from the perspective of an SDN controller. Bianco *et al.* [16] compared the performance of OpenFlow system with link layer Ethernet switching, and network layer IP routing. The authors used the forwarding throughput and the packet latency as major performance indicators.

The buffer behavior has been modeled in several existing works. Luan [17] analyzed buffer behavior in data center networks. Manoj *et al.* [18] analyzed the performance of a buffer-aided multihop relaying system having multiple clusters of relays with buffers using Markov chain. Lai *et al.* [19] proposed a multimedia streaming approach for SDN-enabled 5G network, while considering the mobility and buffer availability information. Sharma *et al.* [20] studied the effect of buffer size in opportunistic networks. Average buffer size in intermittently connected networks has been estimated by Cello *et al.* [21]. Similarly, there is a need for queuing theory-based Markovian analysis of an OpenFlow switch with limited buffer size in OpenFlow systems.

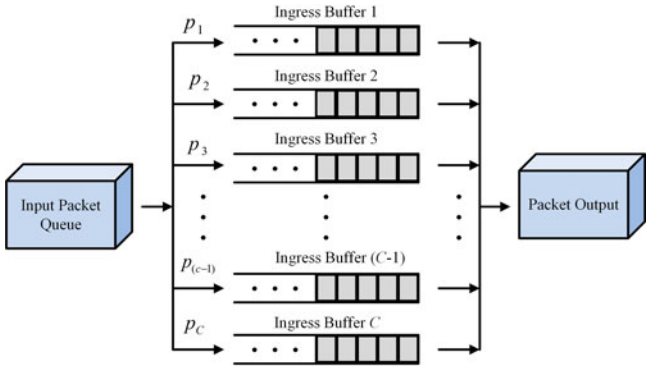

Fig. 1. OpenFlow switch with C ingress ports/buffers.

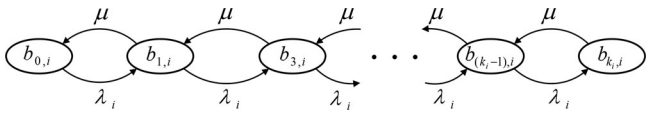

Fig. 2. Ingress port/buffer i of an OpenFlow switch.

Synthesis: We infer that there exist a few research works on performance modeling of an OpenFlow-enabled network. Additionally, some of the works explore different aspects of an OpenFlow switch-based system. Although the buffer behavior has been analyzed in several domains, there is a need to model buffer behavior of an OpenFlow switch. Additionally, an analytical model for buffer size bound evaluation of an OpenFlow switch is in demand.

III. System Model

In this section, we present the architecture of an OpenFlow switch-based system in SDNs. According to the OpenFlow specification version 1.5.0 [6], in an OpenFlow switch, there exist multiple ingress and output ports. Each incoming packet is directed to an ingress port based on the port number embedded in the packet. We consider that there are C ingress ports in an OpenFlow switch, as shown in Fig. 1. Each ingress port $i \in [1, C]$ has a fixed size of buffer, which is denoted as K_i, as shown in Fig. 2. At ingress port i, the mean packet arrival rate and the mean packet processing rate are denoted as λ_i and μ_i, respectively. Hence, the traffic intensity of buffer i in an OpenFlow switch is defined as $\frac{\lambda_i}{\mu_i}$. The packets from each buffer are processed against the same set of flow-rules. According to the OpenFlow specification version 1.5.0, multiple flow-tables exist in an OpenFlow switch. After reaching an OpenFlow switch, each packet gets forwarded to one of the available ingress ports, e.g., ingress port i. Thereafter, it gets queued in the buffer of the ingress port i, i.e., queued at state $b_{m,i}$, where $m \in [0, K_i)$. We consider that packet processing at an OpenFlow switch follows *Markovian Process*.

A. Markovian Process: The Justification

We studied the behavior of an OpenFlow switch using *Markovian model* [5], as it follows the following Markov properties:

1) Each packet is processed individually, and the behavior of an OpenFlow switch is *memoryless*.
2) Packet processing in an OpenFlow switch is a stochastic process having Markov properties, as the conditional probability distribution of the future state depends only on the present state, not on the series of states followed in past.

We consider that a packet gets queued at state X_0, and follows the sequence $X_0 \rightarrow X_1 \rightarrow \cdots \rightarrow X_t$, where X_t defines the state of the packet at time instant t. Therefore, at time instant $(t+1)$, the probability of the packet to be in state X_{t+1} is defined as $P[X_{t+1}|X_t, X_{t-1}, \ldots, X_2, X_1, X_0] = P[X_{t+1}|X_t]$, where X_t, X_{t-1}, and X_{t+1} are the present, immediate past, and future state of a Markovian process, respectively. We consider that the packet arrival rate and the time between arrivals follow Poisson distribution and exponential distribution, respectively, as given in Theorem 1.

Theorem 1: Considering that the packet processing at an OpenFlow switch follows the Markovian process, the arrival of packets and the time between arrivals follow Poisson distribution and exponential distribution, respectively.

Proof: Motivated by *Chapman–Kolmogorov* dynamics [5], we consider that in an infinitesimal time duration $(t, t+\Delta t)$, the mutually exclusive and exhaustive events may occur such as 1) one packet arrives to the buffer of an OpenFlow switch, 2) one packet gets processed and no packet arrival in an OpenFlow switch, and 3) the number of packets in the buffer remains the same. Based on these events, the rate of change in packet flow, $\frac{dp_{m,i}}{dt}$, at mth state of buffer i is defined as follows:

$$\frac{dp_{m,i}(t)}{dt} = \begin{cases} -(\lambda_{m,i} + \mu_{m,i})p_{m,i}(t) + \lambda_{(m-1),i}p_{(m-1),i}(t) \\ +\mu_{(m+1),i}p_{(m+1),i}(t), \quad \text{if } m \geq 1 \\ -\lambda_{0,i}p_{0,i}(t) + \mu_{1,i}p_{1,i}(t), \quad \text{if } m = 0 \end{cases}. \tag{1}$$

By solving (1), we get that the probability of packet getting queued at the ingress buffer of an OpenFlow switch follows the Poisson distribution. On the other hand, the time between arrivals follows the exponential distribution.

B. Packet Flow Through an OpenFlow Switch

Initially, the incoming packets get queued at the ingress buffers. Thereafter, from $b_{0,i}$ of the buffer, each packet enters the ingress flow-table for rule matching, as mentioned in OpenFlow version 1.5.0 [6]. If there is a table hit, then the packet follows the action mentioned in the corresponding matched rule. The action can be one of these— 1) the packet goes to another ingress flow-table having higher index than the current index of the ingress flow-table; 2) the packet enters an egress flow-table, if the egress flag of the packet is set; and 3) the packet goes to the output port or gets dropped, according to the action mentioned in the matched flow rule.

In case of table miss, the action can be one of these—1) the packet gets forwarded to the next flow-table, 2) the packets get forwarded to the controller, according to the table-miss entry, and 3) the packet may get dropped, if either the action in the miss flow entry is to drop packet, or there is no table-miss entry.

We consider that, on an average, each packet gets processed by an OpenFlow switch in $\frac{1}{\mu_i}$ time units. Based on this observation, we model the buffer at each ingress port of an OpenFlow switch as the M/M/1/K/∞ model. Additionally, we consider that there are C ingress ports in an OpenFlow switch. Therefore, in OPUS, we present the queuing model in an OpenFlow switch as C-M/M/1/K/∞ Queuing Model, which is discussed in Section IV.

IV. OPUS Scheme: C-M/M/1/K/∞ Queue

In OPUS, we consider that there are C number of ingress ports. In other words, there are C number of ingress buffers. Each packet can be queued at any of the C ingress buffers based on the ingress port mentioned in the packet. We consider that a packet can be forwarded to the ingress buffer i with probability p_i. Therefore, we get — $\sum_{i=1}^{C} p_i = 1$.

We consider that K_i denotes the size of the buffer i. We define each position in the buffer as a state. Therefore, the mth state of buffer i means the mth position of buffer i. At any infinitesimal time interval dt, the packets can enter the state m of buffer i in two distinct events—1) new packets are added to the queue at the arrival rate of $\lambda_{(m-1),i}$, i.e., state transition follows $[b_{(m-1),i} \rightarrow b_{m,i}]$ at a rate of $\lambda_{(m-1),i}$ and 2) queued packets get processed by the system at a rate of $\mu_{(m+1),i}$, i.e., state transition follows $[b_{(m+1),i} \rightarrow b_{m,i}]$ at a rate of $\mu_{(m+1),i}$. Therefore, in dt time interval, the incoming packet flow, $\texttt{IP}_{m,i}$, in the mth state of buffer i is defined as $\texttt{IP}_{m,i} = p_{(m-1),i}\lambda_{(m-1),i}dt + p_{(m+1),i}\mu_{(m+1),i}dt$.

On the other hand, at infinitesimal time interval dt, the packets can leave from the state m of buffer i in two distinct events: 1) new packets are added to the queue at the arrival rate of $\lambda_{m,i}$. State transition follows $[b_{m,i} \rightarrow b_{(m+1),i}]$ at the rate of $\lambda_{m,i}$ and 2) queued packets get processed by the system at the rate of $\mu_{m,i}$, i.e., state transition follows $[b_{m,i} \rightarrow b_{(m-1),i}]$ at a rate of $\mu_{m,i}$. Therefore, outgoing packet flow, $\texttt{OP}_{m,i}$, in dt time interval from mth state of buffer i is defined as $\texttt{OP}_{m,i} = p_{m,i}\lambda_{m,i}dt + p_{m,i}\mu_{m,i}dt$. Hence, considering that $\frac{dp_{m,i}}{dt} = 0$, we get

$$g_{(m-1),i} = g_{m,i} = \text{constant} \tag{2}$$

where $g_{m,i} = p_{m,i}\lambda_{m,i} - p_{(m+1),i}\mu_{(m+1),i}$. Therefore, from (1), we get

$$p_{m,i} = p_{0,i} \prod_{j=0}^{m-1} \frac{\lambda_{j,i}}{\mu_{(j+1),i}} \quad \forall m \in (0, K_i]. \tag{3}$$

As per OpenFlow specification version 1.5.0 [6], the packets from different ingress buffers get matched against the flow-table entries, simultaneously. Therefore, the packet processing rate of the system is fixed for an OpenFlow switch-based system. The packet processing rate of buffer i is denoted as follows:

$$\mu_{m,i} = \mu \quad \forall m \in [0, K_i] \text{ and } \quad \forall i \in [1, C] \tag{4}$$

where μ is a constant for an OpenFlow switch-based system. On the other hand, the packet arrival rate varies for each buffer i. Therefore, the packet arrival rate for buffer i is denoted as follows:

$$\lambda_{m,i} = \begin{cases} \lambda_i & \forall m \in [0, K_i) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where λ_i is the packet arrival rate for buffer i. Additionally, according to the Poisson's splitting rule [22], we get $\sum_{i=1}^{C} \lambda_i = \lambda$ and $\lambda_i = p_i\lambda$. Based on (4) and (5), the probability of an arrived packet to be at the mth state of buffer i, $p_{m,i}$, is redefined as follows:

$$p_{m,i} = p_{0,i} \prod_{j=0}^{m-1} \frac{\lambda_i}{\mu} = p_{0,i} \left(\frac{\lambda_i}{\mu}\right)^m. \tag{6}$$

Therefore, considering $\sum_{m=0}^{K_i} p_{m,i} = p_i$, from (6), we evaluate the probability of an arrived packet to be in the 0th state of buffer i, i.e., $p_{0,i}$, as follows:

$$p_{0,i} = p_i \left[\frac{1-\frac{\lambda_i}{\mu}}{1-\left(\frac{\lambda_i}{\mu}\right)^{K_i+1}}\right]. \tag{7}$$

Using OPUS, we measure the performance of an OpenFlow switch-based system based on the following parameters: 1) expected number of packets in the system associated with an ingress buffer i of an OpenFlow switch; 2) expected number of packets queued at an ingress buffer i of an OpenFlow switch; 3) expected waiting time of a packet in the system of an OpenFlow switch; and 4) expected waiting time of a packet in buffer i of an OpenFlow switch.

The expected number of packets in the system for buffer i, denoted by $\mathcal{L}_{s,i}$, is expressed as follows:

$$\mathcal{L}_{s,i} = \sum_{m=0}^{K_i} m p_{m,i} = p_i \left(\frac{\lambda_i}{\mu}\right) \left[\frac{1-\left[1+K_i\left(1-\frac{\lambda_i}{\mu}\right)\right]\left(\frac{\lambda_i}{\mu}\right)^{K_i}}{\left(1-\frac{\lambda_i}{\mu}\right)\left[1-\left(\frac{\lambda_i}{\mu}\right)^{K_i+1}\right]}\right]. \tag{8}$$

The expected number of packets in the buffer i is denoted by $\mathcal{L}_{q,i}$. The expected number of packets in service, which is the number of packets getting matched against the ingress and egress flow-table entries, is denoted as $\mathcal{L}_{pr,i}$. We calculate $\mathcal{L}_{pr,i}$ as the probability that the processing unit of OpenFlow switch is busy, and expressed as $\mathcal{L}_{pr,i} = p_i\left(\frac{\lambda_i}{\mu}\right)$.

Therefore, $\mathcal{L}_{q,i}$ is expressed as follows:

$$\mathcal{L}_{q,i} = p_i \left(\frac{\lambda_i}{\mu}\right) \left[\frac{\frac{\lambda_i}{\mu}}{1-\frac{\lambda_i}{\mu}} - \frac{(K_i+1)\left(\frac{\lambda_i}{\mu}\right)^{K_i}}{1-\left(\frac{\lambda_i}{\mu}\right)^{K_i+1}}\right]. \tag{9}$$

Based on (9), the maximum buffer size and the maximum packet traffic intensity of buffer i of an OpenFlow switch are evaluated in Theorems 2 and 3, respectively.

Theorem 2: For a fixed packet traffic intensity, $\frac{\lambda_i}{\mu}$, of buffer i, the maximum buffer size, K_i, needs to satisfy the following constraint:

$$(K_i+1)\ln\left(\frac{\lambda_i}{\mu}\right) + 1 = \left(\frac{\lambda_i}{\mu}\right)^{K_i+1}. \tag{10}$$

Proof: Considering that traffic intensity is fixed, we need to evaluate the maximum buffer size $\mathcal{L}_{q,i}^{\max}$ required for minimizing the packet drop rate. Mathematically

$$\mathcal{L}_{q,i}^{\max} = \max_{K_i} \mathcal{L}_{q,i}. \tag{11}$$

Hence, we take the first-order derivative of $\mathcal{L}_{q,i}$ with respect to K_i and put $\frac{d\mathcal{L}_{q,i}}{dK_i} = 0$. Thereafter, considering that $\frac{\lambda_i}{\mu} \neq 1$ and $p_i \neq 0$, we get (10).

Additionally, performing second-order derivative of $\mathcal{L}_{q,i}$ with respect to K_i, we get that $(1-(\frac{\lambda_i}{\mu})^{K_i+1}) > 0$ and $\frac{d^2\mathcal{L}_{q,i}}{dK_i^2} < 0$. Hence, we argue that for a fixed packet arrival intensity, the maximum value of K_i holds the constraint mentioned in (10). ■

Theorem 3: For a fixed buffer size, K_i, of buffer i, the maximum packet traffic intensity, $\frac{\lambda_i}{\mu}$, needs to satisfy the following constraint:

$$\left[\frac{1-\left(\frac{\lambda_i}{\mu}\right)^{k_i+1}}{1-\left(\frac{\lambda_i}{\mu}\right)}\right]^2 = (k_i+1)^2\left[\frac{\left(\frac{\lambda_i}{\mu}\right)^{k_i-1}}{2-\left(\frac{\lambda_i}{\mu}\right)}\right]. \tag{12}$$

Proof: Considering that buffer size is fixed, our objective is

$$\max \mathcal{L}_{q,i} \tag{13}$$

while satisfying the constraint that $\frac{\lambda_i}{\mu} < 1$. Hence, taking the first-order derivative of $\mathcal{L}_{q,i}$ with respect to $\frac{\lambda_i}{\mu}$ and considering $\frac{d\mathcal{L}_{q,i}}{d\left(\frac{\lambda_i}{\mu}\right)} = 0$, $\frac{\lambda_i}{\mu} \neq 0$, and $p_i \neq 0$, we get the condition for optimum value of $\frac{\lambda_i}{\mu}$ as mentioned in (12).

Additionally, performing the second-order derivative of $\mathcal{L}_{q,i}$ with respect to K_i, we argue that the maximum packet traffic intensity, $\frac{\lambda_i}{\mu}$, follows the constraint given in (12), while taking into consideration that the following inequality holds:

$$(K_i+1)^2\sqrt{\frac{\left(\frac{\lambda_i}{\mu}\right)^{k_i-1}}{2-\frac{\lambda_i}{\mu}}} \leq K_i + (K_i+2)\left(\frac{\lambda_i}{\mu}\right)^{k_i+1}. \tag{14}$$

We define the waiting time of a packet in an OpenFlow switch as the time unit spent by a packet in an OpenFlow switch before leaving the output port. Therefore, the expected waiting time of a packet directed to buffer i in an OpenFlow switch, $\mathcal{W}_{s,i}$, is defined as $\mathcal{W}_{s,i} = \frac{\mathcal{L}_{s,i}}{\lambda_i}$.

We define the waiting time at buffer i as the time unit spent by a packet in an OpenFlow switch before entering into the ingress flow-table. The expected waiting time of a packet at buffer i of an OpenFlow switch, $\mathcal{W}_{q,i}$, is defined as $\mathcal{W}_{q,i} = \frac{\mathcal{L}_{q,i}}{\lambda_i}$.

V. Case Study

We considered two cases: 1) single ingress port, i.e., $C = 1$ and 2) C ingress ports with equal packet traffic intensity, where $C \geq 2$. These cases are discussed briefly in the following section.

A. Case I : $C = 1$

We consider that in an OpenFlow switch, there is a single ingress port. In other words, the number of ingress buffer is

1, i.e., $C = 1$. Hence, the expected number of packets in the system, $\mathcal{L}_{s,1}$, and in the buffer, $\mathcal{L}_{q,1}$, are as follows:

$$\mathcal{L}_{s,1} = \left(\frac{\lambda}{\mu}\right)\left[\frac{1-\left[1+K\left(1-\frac{\lambda}{\mu}\right)\right]\left(\frac{\lambda}{\mu}\right)^{K}}{\left(1-\frac{\lambda}{\mu}\right)\left[1-\left(\frac{\lambda}{\mu}\right)^{K+1}\right]}\right] \tag{15}$$

$$\mathcal{L}_{q,1} = \left(\frac{\lambda}{\mu}\right)\left[\frac{\frac{\lambda}{\mu}}{1-\frac{\lambda}{\mu}} - \frac{(K+1)\left(\frac{\lambda}{\mu}\right)^{K}}{1-\left(\frac{\lambda}{\mu}\right)^{K+1}}\right] \tag{16}$$

where λ and μ are the average packet arrival rate and service rate, respectively, of an OpenFlow switch, and K defines the buffer size of the ingress port. Additionally, the expected waiting time of a packet in an OpenFlow switch before reaching the output port, $\mathcal{W}_{s,1}$, and the waiting time at ingress buffer of an OpenFlow switch, $\mathcal{W}_{q,1}$, are defined as follows:

$$\mathcal{W}_{s,1} = \frac{\mathcal{L}_{s,1}}{\lambda} \text{ and } \mathcal{W}_{q,1} = \frac{\mathcal{L}_{q,1}}{\lambda}. \tag{17}$$

B. Case II : $C \geq 2$

We consider that there are at least two ingress ports ($C \geq 2$) in an OpenFlow buffer. Additionally, we consider that each arrived packet has an equal probability of being at any of the available ingress port or buffer. Therefore, the probability of a packet being queued at buffer i, p_i, is $\frac{1}{C}$

$$\mathcal{L}_{s,i} = \frac{\lambda}{C^2\mu}\left[\frac{1-\left[1+K\left(1-\frac{\lambda}{C\mu}\right)\right]\left(\frac{\lambda}{C\mu}\right)^{K}}{\left(1-\frac{\lambda}{C\mu}\right)\left[1-\left(\frac{\lambda}{C\mu}\right)^{K+1}\right]}\right] \tag{18}$$

$$\mathcal{L}_{q,i} = \frac{\lambda}{C^2\mu}\left[\frac{\frac{\lambda}{C\mu}}{1-\frac{\lambda}{C\mu}} - \frac{(K+1)\left(\frac{\lambda}{C\mu}\right)^{K}}{1-\left(\frac{\lambda}{C\mu}\right)^{K+1}}\right] \tag{19}$$

where $\lambda_1 = \cdots = \lambda_C = \frac{\lambda}{C}$ and $K_1 = \cdots = K_C = K$. Hence, λ and K are constants for a specific OpenFlow switch. Additionally, the expected waiting time of a packet in an OpenFlow switch before reaching output port, $\mathcal{W}_{s,i}$, and the waiting time at ingress buffer i of an OpenFlow switch, $\mathcal{W}_{q,i}$, are defined as follows:

$$\mathcal{W}_{s,i} = \mathcal{L}_{s,i}\frac{C}{\lambda} \quad \text{and} \quad \mathcal{W}_{q,i} = \mathcal{L}_{q,i}\frac{C}{\lambda} \tag{20}$$

VI. Performance Evaluation

In this section, we analyze the required buffer size of an OpenFlow switch-based system with varying packet arrival and processing rate in SDN. We evaluate the performance of the proposed OPUS scheme for an OpenFlow switch, based on the parameters such as maximum arrival rate, minimum buffer size, and maximum waiting time. Generic test-bed information for OPUS is provided in Table I. For simplicity, we evaluate the buffer size requirement of a single OpenFlow switch in SDN. Additionally, we consider that each packet gets queued at buffer $i \in [1, C]$ of an OpenFlow switch with a probability p_i. We consider that each packet selects a buffer, i.e., queue, i, randomly.

1) Simulation Parameters: We simulated the performance of an OpenFlow switch-based system in SDN, where each OpenFlow switch has multiple number of queues such as 2 and 6, as mentioned in Table II. The total size of buffer for each OpenFlow switch is varied in 0.5–1 million packets (mp). On the other hand, the packet processing rate is varied in 0.01–0.05 million packets per seconds (mpps), as mentioned in Table II. The size of each packet is considered as 1500 B[15]. We simulate OPUS for different simulation durations: 25, 50, 75, 100 ms.

TABLE I
System Specification

Parameter	Value
Processor	Intel(R) Core(TM) i5-2500 CPU @ 3.30 GHz
RAM	4 GB DDR3
Disk Space	500 GB
Operating System	Ubuntu 16.04 LTS
Application Software	MATLAB 2015b

TABLE II
Simulation Parameters

Parameter	Value
Number of OpenFlow switch	1
Number of buffers	$2, 6, 10$
Total buffer size	$0.5, 0.75, 1$ million packets
Packet arrival rate	$0.15, 0.20, 0.25$ $mpps$
Packet processing rate	$0.01, 0.025, 0.05$ $mpps$
Packet size	1500 $Byte$ [15]
Simulation Duration	100 ms

2) Performance Metrics: We evaluated the performance of an OpenFlow switch with different number of queues or buffers such as $2, 6, 10$, in the proposed C-M/M/1/K/∞ queue-based scheme, OPUS, while considering the following parameters:

Maximum Arrival Rate (ArrRate): The maximum arrival rate depends on the average buffer size and the maximum processing rate. It varies proportionally with the average buffer size of an OpenFlow switch. Additionally, the maximum arrival rate varies proportionally with the number of buffers and the processing rate of an OpenFlow switch.

Minimum Buffer Size (BuffSize): The OpenFlow switches have ternary content-addressable memory (TCAM), which are costly. Hence, we need to evaluate the minimum buffer size requirement of an OpenFlow switch for an optimum traffic intensity.

Maximum Waiting Time: The performance of an OpenFlow switch mostly depends on the waiting time of a packet in the system. With the increase in the waiting time, the performance of an OpenFlow switch degrades. Hence, we need to evaluate the maximum waiting time of a packet in an OpenFlow switch.

3) Results and Discussions: For simulation, we considered that the packets enter through the ingress port of an OpenFlow switch, and get forwarded *randomly* to any of the available buffers, before getting matched against the ingress flow-tables. Thereafter, based on the table-hit and table-miss entry, the packets are processed. Additionally, we consider that the packets, which are forwarded to the SDN controller, get queued in the ingress buffers as newly arrived packets.

From Fig. 3, we observe that the maximum arrival rate, which can be handled by an OpenFlow switch, increases by

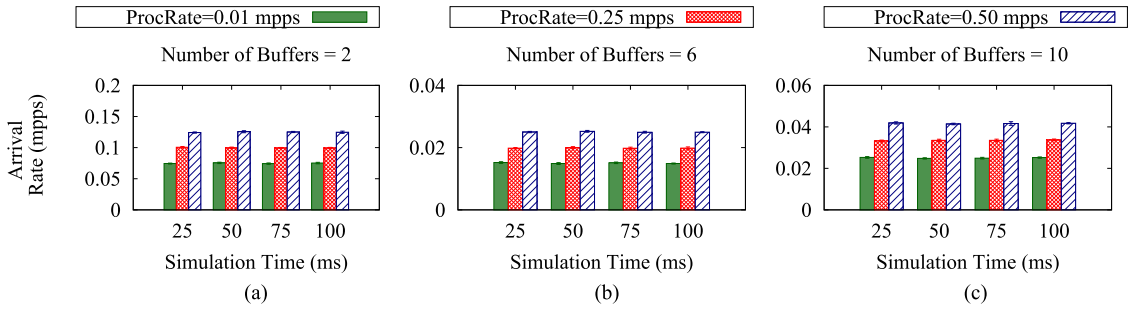

Fig. 3. Maximum arrival rate per OpenFlow switch.

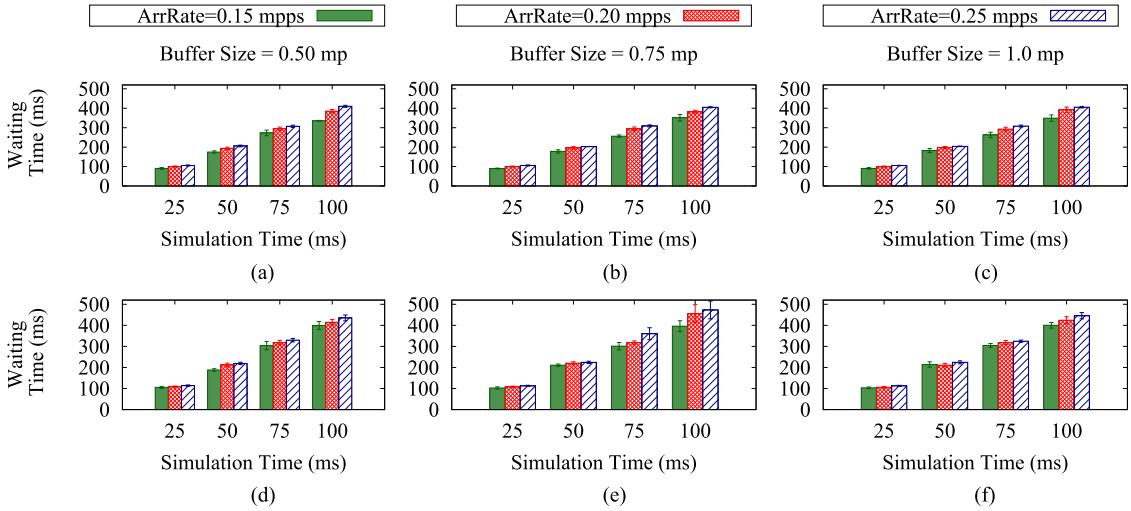

Fig. 4. Maximum waiting time per OpenFlow switch. In (a), (b), and (c), and in (d), (e), and (f), number of buffers are 2 and 6, respectively. Buffer size per queue is 0.50, 0.75, and 1.0 mp in (a) and (d), (b) and (e), and (c) and (f), respectively.

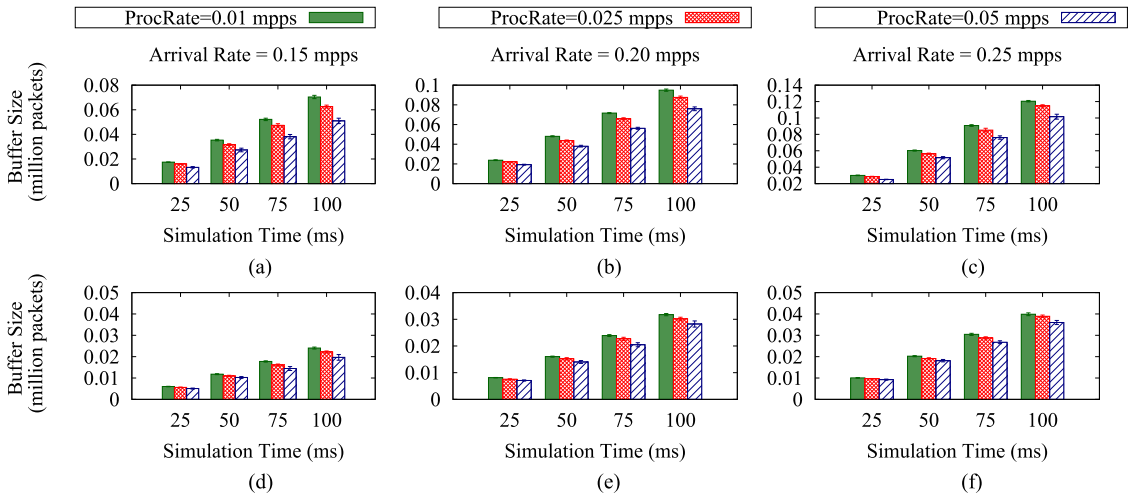

Fig. 5. Minimum buffer size per OpenFlow ingress port. In (a), (b), and (c), and in (d), (e), and (f), number of buffers are 2 and 6, respectively. Packet arrival rates are 0.15, 0.20, and 0.25 mpps in (a) and (d), (b) and (e), and (c) and (f), respectively.

26.15%–30.4% with the increase in processing rate of an OpenFlow switch by *two times*. Additionally, we observe that with the increase in the number of buffers, the maximum packet arrival rate per buffer decreases. However, the maximum packet arrival rate in an OpenFlow switch remains the same, while considering that the buffer size of an OpenFlow switch is constant. Therefore, we conclude that arrival rate of an OpenFlow switch remains constant with fixed buffer size and fixed processing rate.

From Fig. 5, we observe that with the increase in the arrival rate, the minimum buffer size requirement increases. On the

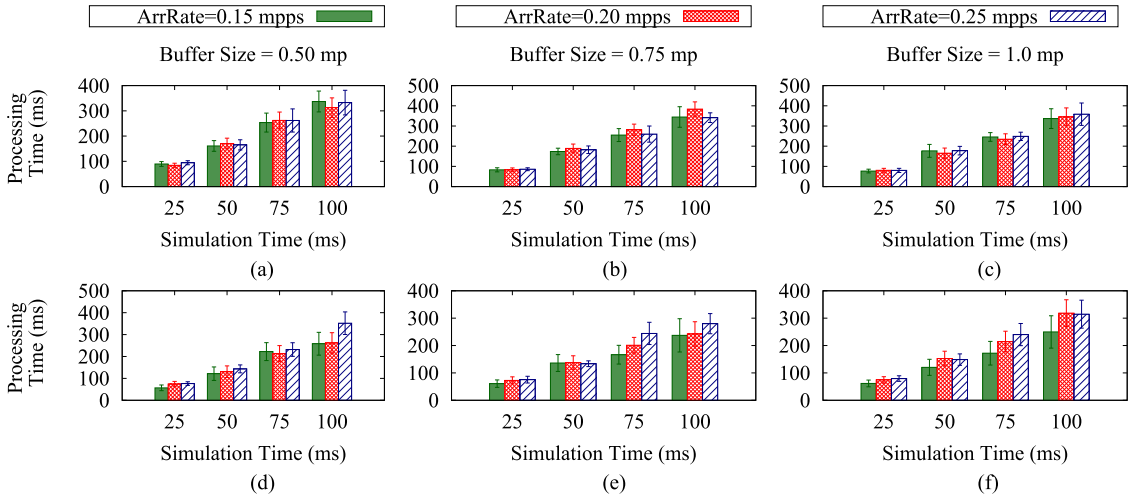

Fig. 6. Maximum processing time per OpenFlow switch. In (a), (b), and (c), and in (d), (e), and (f), number of buffers are 2 and 6, respectively. Buffer size per queue are 0.50, 0.75, and 1.0 mp in (a) and (d), (b) and (e), and (c) and (f), respectively.

other hand, for fixed arrival rate, with the increase in processing rate, the buffer size requirement increases up to the processing rate 0.03–0.35 mpps. Thereafter, the minimum buffer size remains constant. Hence, we conclude that the optimum arrival rate and processing rate of an OpenFlow switch lie in the ranges 0.20–0.25 and 0.03–0.35 mpps, respectively. The minimum buffer size required is in the range 0.67–0.80 mp.

Fig. 6 shows that for buffer size 0.75 mp, the processing rate of an OpenFlow switch is less, while considering that the packet arrival rate varies in the range of 0.15–0.25 mpps. On the other hand, from Fig. 6, we infer that the packet processing rate of an OpenFlow switch varies insignificantly. Hence, we conclude that for the packet arrival rate in an OpenFlow switch with rate 0.15–0.25 mpps, the optimum number of buffers is *2*. Additionally, the optimum buffer size of an OpenFlow switch is 0.75 mp, i.e., 1.125 GB, while considering that each packet is of size 1500 B, as mentioned in Table II. In Fig. 4, we observe that the packet waiting time is less for an OpenFlow switch with buffer size 0.50 mp, as few packets get dropped due to insufficient buffer space at an OpenFlow switch. On the other hand, in Fig. 4, we observe that the waiting time of an OpenFlow is similar for OpenFlow switches with buffer size 0.75 and 1.00 mp. Hence, we conclude that the minimum waiting time at an OpenFlow switch can be ensured with buffer size of 0.75 mp, i.e., 1.125 GB. These analytical results confirm with the OpenFlow specification given in [15] and [23].

We maintain that the performance of an OpenFlow switch can be improved with the packet arrival and processing rate of 0.20–0.25 and 0.03–0.35 mpps, respectively. On the other hand, the optimum buffer size of an OpenFlow switch is 0.75 mp, i.e., 1.125 GB, as observe using OPUS.

VII. Conclusion

In this paper, we analyzed the optimum buffer size of an OpenFlow switch in order to ensure quality-of-service in OpenFlow systems. We analyzed the optimum packet arrival and processing rates and the average waiting of packets in an OpenFlow switch-based system. In the proposed scheme, we modeled the architecture of an OpenFlow switch as a C-M/M/1/K/∞ queue, while considering that there are C ingress buffers. Each buffer has K memory blocks in an OpenFlow switch. We analyzed the optimum number of buffers with the optimum value of each buffer. Additionally, we evaluated the optimum packet arrival and processing rates of an OpenFlow switch using the proposed scheme, OPUS, in OpenFlow systems.

Future extension of this work is to design a scheme for improving the queuing model with multiple OpenFlow switches, and reducing waiting time or queuing delay in an OpenFlow system, while ensuring proper utilization of TCAM memory in an OpenFlow switch-based system. This work also can be extended to visualize using SDN emulator such as Mininet, while considering real-time parameters—queuing delay for interswitch communication and duration for flow-table update. In addition, this work can be extended to understand how the queuing model for group table functions in an OpenFlow switch-based system with proper utilization of TCAM memory.

References

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

[2] P. T. Congdon, P. Mohapatra, M. Farrens, and V. Akella, "Simultaneously reducing latency and power consumption in OpenFlow switches," *IEEE/ACM Trans. Netw.*, vol. 22, no. 3, pp. 1007–1020, Jun. 2014.

[3] N. P. Katta, J. Rexford, and D. Walker, "Incremental consistent updates," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 49–54.

[4] C. R. Meiners, A. X. Liu, and E. Torng, "Bit weaving: A non-prefix approach to compressing packet classifiers in TCAMs," *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 488–500, Apr. 2012.

[5] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. New York, NY, USA: Wiley-Interscience, 1975.

[6] OpenFlow. (2014, Dec.) OpenFlow Switch Specification Version 1.5.0. Open Networking Foundation. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/o nf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf

[7] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee, "DevoFlow: Cost-effective flow management for high performance enterprise networks," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw.*, 2010, pp. 1–6.

[8] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for Network Update," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2012, pp. 323–334.

[9] M.-H. Wang, P.-W. Chi, J.-W. Guo, and C.-L. Lei, "SDN storage: A stream-based storage system over software-defined networks," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 598–599.

[10] F. Li, J. Cao, X. Wang, Y. Sun, T. Pan, and X. Liu, "Adopting SDN switch buffer: Benefits analysis and mechanism design," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. Workshops*, Jun. 2017, pp. 2171–2176.

[11] S. Bera, S. Misra, S. K. Roy, and M. S. Obaidat, "Soft-WSN: Software-defined WSN management system for IoT applications," *IEEE Syst. J.*, pp. 1–8, Nov. 2016, DOI: 10.1109/JSYST.2016.2615761.

[12] M. Hayes, B. Ng, A. Pekar, and W. K. G. Seah, "Scalable architecture for SDN traffic classification," *IEEE Syst. J.*, pp. 1–12, Apr. 2017, DOI: 10.1109/JSYST.2017.2690259.

[13] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an OpenFlow architecture," in *Proc. 23rd Int. Teletraffic Congr.*, 2011, pp. 1–7.

[14] C. Metter, M. Seufert, F. Wamser, T. Zinner, and P. Tran-Gia, "Analytical model for SDN signaling traffic and flow table occupancy and its application for various types of traffic," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 3, pp. 603–615, Sep. 2017.

[15] S. Azodolmolky, R. Nejabati, M. Pazouki, P. Wieder, R. Yahyapour, and D. Simeonidou, "An analytical model for software defined networking: A network calculus-based approach," in *Proc. IEEE Global Commun. Conf.*, Dec. 2013, pp. 1397–1402.

[16] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "OpenFlow switching: Data plane performance," in *Proc. IEEE Int. Conf. Commun.*, May 2010, pp. 1–5.

[17] G. Luan, "Buffer stopping time analysis in data center networks," *IEEE Commun. Lett.*, vol. 18, no. 10, pp. 1739–1742, Oct. 2014.

[18] B. R. Manoj, R. K. Mallik, and M. R. Bhatnagar, "Buffer-aided multi-hop DF cooperative networks: A state-clustering based approach," *IEEE Trans. Commun.*, vol. 64, no. 12, pp. 4997–5010, Dec. 2016.

[19] C. F. Lai, Y. C. Chang, H. C. Chao, M. S. Hossain, and A. Ghoneim, "A buffer-aware QoS streaming approach for SDN-enabled 5G vehicular networks," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 68–73, 2017.

[20] D. K. Sharma, S. K. Dhurandher, I. Woungang, R. K. Srivastava, A. Mohananey, and J. J. P. C. Rodrigues, "A machine learning-based protocol for efficient routing in opportunistic networks," in *IEEE Syst. J.*, Dec. 2016, pp. 1–7, DOI: 10.1109/JSYST.2016.2630923.

[21] M. Cello, G. Gnecco, M. Marchese, and M. Sanguineti, "A model of buffer occupancy for ICNs," *IEEE Commun. Lett.*, vol. 16, no. 6, pp. 862–865, Jun. 2012.

[22] N. Sapountzis, T. Spyropoulos, N. Nikaein, and U. Salim, "Optimal downlink and uplink user association in backhaul-limited HetNets," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[23] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore, "OFLOPS: An open framework for OpenFlow switch evaluation," in *Proc. Int. Conf. Passive Active Netw. Meas.*, 2012, pp. 85–95.

Author's photographs and biographies not available at the time of publication.