AMOPE: Performance Analysis 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 defining probabilistic bounds of packet flow through an OpenFlow switch in software-defined networks (SDNs). The problem is challenging, as OpenFlow is one of the popular southbound application programming interfaces, which enables controller-switch interaction. The related existing literature addresses the different aspects of OpenFlow and SDN-controller interactions. However, there is a need to analyze the performance of the OpenFlow switch, in order to determine the bounds of the performance measures. In this paper, we propose Markov chain-based analytical model, named AMOPE, for analyzing packet flow through an OpenFlow switch, while defining the probabilistic bounds on performance analysis. Additionally, in AMOPE, we propose a state diagram based on the OpenFlow specification version 1.5.0, and calculate the theoretical probabilities of a packet to be in different states of the OpenFlow switch. Further, AMOPE defines the theoretical bounds of OpenFlow performance measures such as the output action, packet drop, and send to the controller probabilities. Simulation-based analysis exhibits that approximately 60% of the processed packets are sent to output action, 31% of the processed packets are sent to the controller, and the remaining processed packets are dropped in an OpenFlow switch.**

***Index Terms*—OpenFlow, Software-Defined Networks, Markov Chain, Performance Modeling, Packet Flow**

I. Introduction

Software-defined network (SDN) enables in decoupling the network control tasks from the tasks of packet forwarding and processing [1] while dividing into two parts – the *control* and the *data* planes. The control plane includes northbound and southbound application programming interfaces (APIs). Presently, OpenFlow is the most popular southbound API that enables controller-switch interaction in SDN architecture.

An OpenFlow switch contains one or more flow-tables to store packet forwarding rules. There are two types of flow-tables, namely ingress and egress flow-tables. When a packet arrives at the ingress port of an OpenFlow switch, it is matched with the flow entries of the first flow-table and with the entries of subsequent flow-tables, if required. Matched flow entry provides a set of actions to be executed for the corresponding packet. If no matching entry is found, the packet is dropped or forwarded to the controller for installation of new rule depending on the network policy. Each switch communicates with each external controller via an OpenFlow channel.

In the existing literature, the researchers addressed the different aspect of OpenFlow and SDN-controller interactions. Additionally, OpenFlow version 1.5.0 [2] enabled hardware and software switches are considered by the researchers. However, there is a need to analyze the performance of the OpenFlow switch, in order to determine the bounds, i.e., probabilistic bounds, of the performance measures, and to suggest possible improvements. Additionally, researchers have proposed different schemes and architectures for SDN, viz., [1], [3], [4], while considering OpenFlow protocol and switches. The proposed approaches require a substantial amount of time, depending on the available hardware. Hence, there is a need of an analytical model to evaluate the performance of OpenFlow. The analytical model is to be used for evaluating the bounds of performance metrics before the actual implementation or simulation.

In the existing literature, there are very few works that provide analytical model for performance analysis of an OpenFlow-enabled network [5], [6], [7]. However, there is a need of an analytical model to define the probabilistic bounds of the OpenFlow protocol version 1.5.0 [2]. Moreover, there is a need of an evaluation of the network performance, while considering necessary parameters such as throughput, packet processing time, delay, and packet drop count.

In this work, we model packet flow through an OpenFlow switch as a Markov chain and formulate probabilistic expressions for the network parameters. We consider that in the presence of Internet of things (IoT) devices, the number of *mice flows*[8] increases significantly. Therefore, there is a need for packet-centric analysis of OpenFlow SDN. To the best of our knowledge, this is the first Markovian model of SDN architecture that considers the packet-centric analysis of an OpenFlow switch. Our model is also the first one that takes into consideration both the ingress and egress processing of packets based on OpenFlow version 1.5.0 [2]. The primary contributions of our work are summarized below.

1) Initially, an analytical model based on the existing OpenFlow protocol [2] is developed using Markov chain. This model depicts all the three stages of a packet life cycle inside an OpenFlow switch such as waiting in switch queue, ingress processing, and egress processing.

2) We perform a probabilistic analysis on the developed model. Based on the analysis, we comment on different packet flow probabilities such as output action probability, packet drop probability, and send to controller probability.

3) Finally, in a simulated environment, we estimated necessary parameters such as throughput, average delay per packet, packet drop count, and average packet processing time.

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

II. Related Work

Two streams of the existing works are worth reviewing related to the problem of performance improvement in the presence of OpenFlow switches — (a) improvement of the OpenFlow enabled networks [4], [9], and (b) performance modeling of the OpenFlow architecture [5], [6], [10].

Concerning the improvement of the OpenFlow networks, in the existing literature, Reitblatt *et al.* [9] developed a model for OpenFlow networks that allows configuring the entire network in one instance, while taking help of abstract operations. Additionally, Katta *et al.* [3] provided a consistent network update mechanism that addresses the trade-off between rule-space overhead and update time. Meiners *et al.* [4] proposed a compression technique for flow-table entries. Rego *et al.* [11] proposed a vehicular traffic management scheme for emergency situations with the help of SDN. In another scheme, Rego *et al.* [12] proposed a learning-based error correction scheme for SDN-based multimedia networks. Trestian *et al.* [8] proposed a load balancing protocol for OpenFlow-based data center networks (DCNs) with multipath routing. Kampanakis *et al.* [13] studied the application of SDN for ensuring security in mobile devices.

Even though in the existing literature, researchers explored different aspects in the context of OpenFlow and SDN and different performance analytical models in other domains, very few works address performance analysis of an OpenFlow enabled SDN architecture. Javed *et al.* [7] evaluated the latency for OpenFlow SDN considering OpenFlow version 1.0. Salah *et al.* [14] proposed a Markovian analytical model using queuing theory. The authors analyzed the performance of rule-based firewalls in the presence of DoS attacks. Misra *et al.* [15] studied the optimal buffer size of an OpenFlow switch using *C-M/M/1/K* queuing model. Eager and Sevcik [16] studied the performance bounds for single-class queuing networks with fixed rates and delay service centers using mean-value analysis. The authors claimed that the performance bounds ensure the accuracy of the model.

On the other hand, few works in the existing literature, which address the performance analysis of an OpenFlow-enabled SDN architecture, are discussed here. Faraci and Schembra [17] proposed a Markovian analytical model for managing OpenFlow-based SDN customer premises equipment, and evaluated the cost, theoretically. Jarchel *et al.* [6] modeled the OpenFlow architecture as M/M/1 forward and an M/M/1-S feedback queuing systems. This model measures the delay in an OpenFlow switch, and estimates the total sojourn time of a packet and probability of dropped packets. This work is based on OpenFlow version 1.0.0, where each switch has a single flow-table. The authors assumed the queue length of a switch to be infinite. However, according to OpenFlow version 1.5.0 [2], each switch has multiple flow-tables (both ingress and egress) with more number of match fields. In another work, Azodolmolky *et al.* [5] modeled SDN using network calculus. This model analyzes network performance from an SDN controller's perspective. Metter *et al.* [18] developed an analytical model for network performance optimization while considering table-occupancy and singling-rate of an OpenFlow switch. On the other hand, Bianco *et al.* [10] compared the performance of OpenFlow switching with that of link layer Ethernet switching, and network layer IP routing. The authors used the packet latency and the forwarding throughput as major performance indicators.

Synthesis: Thus, we infer that there exist a few works on analytical modeling of OpenFlow switch in the existing literature. Additionally, the researchers explored different aspects of the OpenFlow switch. However, there is a need for an analytical model to determine the bounds of the performance metrics of an OpenFlow switch. Additionally, an analytical model for OpenFlow version 1.5.0 in the presence of ingress and egress flow-tables is in demand.

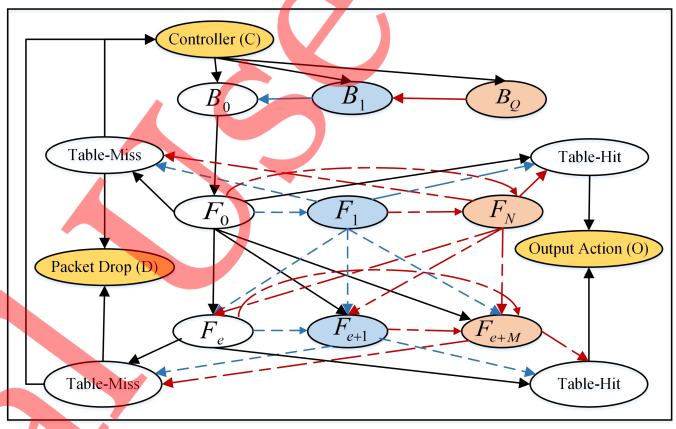


Fig. 1: State Diagram for Packet Flow in an OpenFlow Switch

III. Proposed Analytical OpenFlow Bounds Model

In this work, we develop a Markovian model, named AMOPE, to replicate behavior of an OpenFlow switch based on OpenFlow switch specification version 1.5.0 [2], when an incoming flow of packets passes through it. In AMOPE, we consider multiple switches instead of a single switch per controller. For each switch, we estimate the necessary performance metrics considering packet queuing, ingress and egress processing. The OpenFlow switch considers packet level services. Therefore, in AMOPE, the switch takes each packet as an individual entity, despite taking flow-specific data. In this paper, we present a Markovian analysis of packet flow through an OpenFlow switch using Markov chain [19], [20]. An OpenFlow switch has three parts — the switch queue, and ingress and egress processing units. We describe the state diagram and the probabilistic analysis of each part of an OpenFlow switch in Sections III-B and III-C. In AMOPE, we assume that — (1) each mouse flow comprises a few number of packets, (2) the packet arrival process follows a poison distribution, and (3) packet inter-arrival time follows an exponential distribution.

A. Markovian Model: The Justification

We studied the behavior of an OpenFlow switch using *Markovian model* [21], [22], as it follows the following Markov properties:

1) Each packet is processed individually, i.e., the behavior of an OpenFlow switch is *memoryless*.

2) Packet processing in an OpenFlow switch is a stochastic process having Markov properties, as $P(x_{n+1}|X_n, X_{n-1}, X_{n-2}, \cdots, X_1) = P(x_{n+1}|X_n)$, where X_n, X_{n-1}, and X_{n+1} are the present, immediate past, and future state of a Markovian process, respectively.

B. State Diagram

We consider that when a packet is sent from the controller to the OpenFlow switch, the packet gets queued, initially, before entering the ingress processing unit, as shown in Figure 1. The OpenFlow switch has a queue of length of size $(Q+1)$, and each i^{th} position of the OpenFlow queue is denoted as B_i, where $0 \leq i \leq Q$. If the packet gets queued at B_i, it waits for a finite duration in order to reach the 0^{th} position of the queue, B_0. Thereafter, the packet enters the ingress processing unit of the switch, and searches for a match at the 0^{th} ingress flow-table, F_0. Hence, there can be *table-hit*, signifying a match found in the table, or *table-miss*, when no match is found. In case of table-hit, the OpenFlow switch executes one of the instructions — (1) the packet goes to another ingress table F_j, $j \in (0, N]$ or (2) the action mentioned in the action field of the flow entry gets executed.

On the other hand, in case of table-miss, the packet follows one of the possibilities — (1) pass to an ingress flow-table F_j, where $0 < j \leq N$, (2) pass to the controller, according to the table-miss flow entry, or (3) drop the packet, according to the table-miss flow entry; or (4) drop the packet if there is no table-miss flow entry.

After reaching to the ingress flow-table F_j, the packet is matched against the flow-table entries. If there is table-hit, either the packet gets forwarded to an ingress flow-table $F_{j'}$, where $j' > j$, or instructions are executed according to the flow-table entry, as discussed earlier. On the other hand, in case of table-miss, the packet gets forwarded either to an ingress flow-table $F_{j'}$, where $j' > j$, to the controller, or gets dropped, according to the table-miss flow entry.

After the processing of the packet at the ingress processing unit, and the output action taken according to the table-hit at the ingress flow-table. If the *egress flag* is set, the packet enters the *egress processing unit*. Once the packet enters the egress processing unit, the packet gets forwarded to the egress flow-table F_e. The packet is matched against the flow-table entries. For egress table-hit at $(e+k)^{th}$ egress flow-table F_{e+k}, where $0 \leq k < M$, the packet either gets forwarded to another egress flow-table $F_{e+k'}$, where $k < k' \leq M$, or forwarded for executing action set as mentioned in the action field of the matched entry. On the other hand, in case of table-miss, the packet follows one of the options — (1) the packet gets forwarded to the next egress flow-table $F_{e+k'}$, from the egress flow-table F_{e+k}, where $k < k' \leq M$; (2) the packet is sent to the controller; (3) the packet is dropped, according to the egress table-miss flow entry; or (4) the packet is dropped if there is no egress table-miss entry.

If the packet is sent to the controller, the controller handles the packet and forwards the packet to the available SDN switches, while either making modifications in the flow-table entries or rerouting the packet. Otherwise, the controller also has a provision to drop the packet.

C. Probabilistic Analysis

We consider that SDN is comprised of a single controller unit and multiple OpenFlow switches. In this paper, we focus on packet flow through an OpenFlow switch. We consider that the probability of packet getting forwarded from the controller to the specific switch having queue length $(Q+1)$, where indexing starts from the 0^{th} position, is p'. We consider that the packets get forwarded to any of the available OpenFlow switches without any biases. If there are $\mathbb{S}$ number of OpenFlow switches in the network, we consider that the probability of packet getting forwarded to OpenFlow switch s is defined as $p' = \frac{1}{\mathbb{S}}$. After getting forwarded by the controller, the packet gets queued at the switch buffer. Considering that the queue length is $(Q+1)$, and the packet getting queued at any position of the buffer is unbiased, i.e., equally probable, we get:

$$P(B_i|C) = \frac{p'}{Q+1}, \quad \text{where } 0 \leq i \leq Q \tag{1}$$

where B_i defines the i^{th} position of the buffer, C denotes the SDN-controller, and $P(B_i|C)$ denotes the probability of the packet getting forwarded to the buffer B_i from the controller C. After getting queued at buffer B_i, the packet gets forwarded to the next position of the OpenFlow queue, B_{i-1}, sequentially. Hence, we get — $P(B_{i-1}|B_i) = 1$, where $0 < i \leq Q$. After reaching at B_0, the packet enters the ingress processing unit of the OpenFlow switch, i.e., the first ingress flow-table F_0. Hence, we have — $P(F_0|B_0) = 1$.

We consider that at the ingress flow-table F_j, the packet gets matched against the flow-table entries of F_j. The packet finds either table-hit or table-miss, as discussed in Section III-B. We consider that the probability of having table-hit at the ingress flow-table F_j is p_j. Hence, the probability of table-miss at the ingress flow-table F_j is $(1-p_j)$. Additionally, we consider that in case of table-hit at ingress flow-table F_j, where $0 \leq j < N$, there are three *equally probable* events, i.e., considered to be unbiased events, — (1) the packet gets forwarded to the egress flow-table F_e, (2) the packet is handled according to the output action, and (3) the packet gets forwarded to any of the next ingress flow-tables, $F_{j'}$, where $j < j' \leq N$. Hence, we get:

$$P(F_e|F_j) + P(O|F_j) + \sum_{j<j'\leq N} P(F_{j'}|F_j, \texttt{TH}) = p_j \tag{2}$$

where $P(F_e|F_j)$, $P(O|F_j)$, and $P(F_{j'}|F_j, \texttt{TH})$ define the probability of the packet getting forwarded to the egress table F_e, the probability of packet executed according to the output action, and the probability of the packet getting forwarded to flow-table $F_{j'}$ for table-hit, respectively. In case of table-miss at the ingress flow-table F_j, there are three *equally probable unbiased* events — (1) the packet gets forwarded to any of the next ingress flow-tables $F_{j'}$, where $j < j' \leq N$, (2) the packet gets forwarded to the SDN controller, and (3) the packet gets dropped. We get:

$$P(C|F_j)+P(D|F_j)+\sum_{j<j'\leq N} P(F_{j'}|F_j,\texttt{TM}) = (1-p_j) \quad (3)$$

where $P(F_{j'}|F_j,\texttt{TM})$, $P(C|F_j)$, and $P(D|F_j)$ define the the probability of the packet getting forwarded to the flow-table $F_{j'}$ for table-miss, the probability of packet getting forwarded to the controller, and the probability of the packet getting dropped, respectively. Therefore, the probability $P(F_{j'}|F_j)$ of the packet getting forwarded to a next ingress flow-table $F_{j'}$, where $0 < j < j' \leq N$, and the probability $P(F_e|F_j)$ of the packet getting forwarded to F_e are as follows:

$$P(F_{j'}|F_j) = \frac{1}{3(N-j)} \text{ and } P(F_e|F_j) = \frac{p_j}{3} \quad (4)$$

The packet cannot be forwarded to the ingress flow-table $F_{j'}$ from the ingress flow-table F_j, i.e., $P(F_{j'}|F_j) = 0$, where $0 \leq j' \leq j \leq N$. If the packet finds a table-hit at the flow-table F_N, there are two *equally probable* events without biasness — (1) the packet gets forwarded to the egress flow-table F_e, and (2) the packet is handled according to the output action. Therefore, the probability that the packet gets forwarded to F_e from the ingress flow-table F_N, is given as:

$$P(F_e|F_N) = \frac{p_N}{2} \quad (5)$$

where p_N is the probability of getting a table-hit at the ingress flow-table F_N. Similar to the ingress flow-tables, at the egress flow-table F_{e+k}, we consider that the probability of getting a table-hit is defined as p_{e+k}. In case of table-hit, there are two *equally probable unbiased* outcomes — (1) the packet gets forwarded to any next egress flow-table $F_{e+k'}$, and (2) the packet is handled according to the output action. On the other hand, in case of table-miss with probability $(1-p_{e+k})$, which adds three *equally probable* events such as (1) the packet gets forwarded to any next egress flow-table, $F_{e+k'}$, where $0 < k < k' \leq M$, (2) the packet gets forwarded to the SDN controller, and (3) the packet gets dropped. Hence, the probability $P(F_{e+k'}|F_{e+k})$ of the packet getting forwarded to the next egress flow-table $F_{e+k'}$ from the egress flow-table F_{e+k} is expressed as follows:

$$P(F_{e+k'}|F_{e+k}) = \frac{p_{e+k}}{2(M-k)} + \frac{1-p_{e+k}}{3(M-k)} = \frac{2+p_{e+k}}{6(M-k)} \quad (6)$$

On the other hand, similar to the ingress flow-table rules, the packet cannot be forwarded to any egress flow-table with lower index, i.e., $P(F_{e+k'}|F_{e+k}) = 0$, where $0 \leq k' \leq k \leq M$. From Figure 1, we get that the packet may reach to the *output action* state, denoted as O, form each ingress flow-tables F_j, where $0 \leq j \leq N$, and each egress flow-table F_{e+k}, where $0 \leq k \leq M$, when there is a table-hit. We define the probability $P(O|F_j)$ of the packet reaching to output action state from any flow-table F_j, i.e., either ingress or egress flow-table, is given as follows:

$$P(O|F_j) = \begin{cases} \frac{p_j}{3}, & \text{if } 0 \leq j < N \\ \frac{p_j}{2}, & \text{if } j = N \text{ and } e \leq j < e+M \\ p_j, & \text{if } j = e+M \end{cases} \quad (7)$$

On the other hand, from Figure 1, we observe that the packet may get dropped, where the state is represented as D, form each ingress flow-table F_j, where $0 \leq j \leq N$, and each egress flow-table F_{e+k}, where $0 \leq k \leq M$, in case of table-miss. We define the probability $P(D|F_j)$ of the packet reaching to packet drop state from any flow-table F_j is given as follows:

$$P(D|F_j) = \begin{cases} \frac{1-p_j}{3}, & \text{if } 0 \leq j < N \text{ and } e \leq j < e+M \\ \frac{1-p_j}{2}, & \text{if } j = N \text{ and } j = e+M \end{cases} \quad (8)$$

Similarly, from Figure 1, we get that the packet may be forwarded to the SDN controller, where the state is represented as C, form each ingress flow-table F_j, where $0 \leq j \leq N$, and each egress flow-table F_{e+k}, where $0 \leq k \leq M$, in case of table-miss. We define the probability $P(C|F_j)$ of the packet reaching to packet drop state from any flow-table F_j is given as follows:

$$P(C|F_j) = P(D|F_j) \quad (9)$$

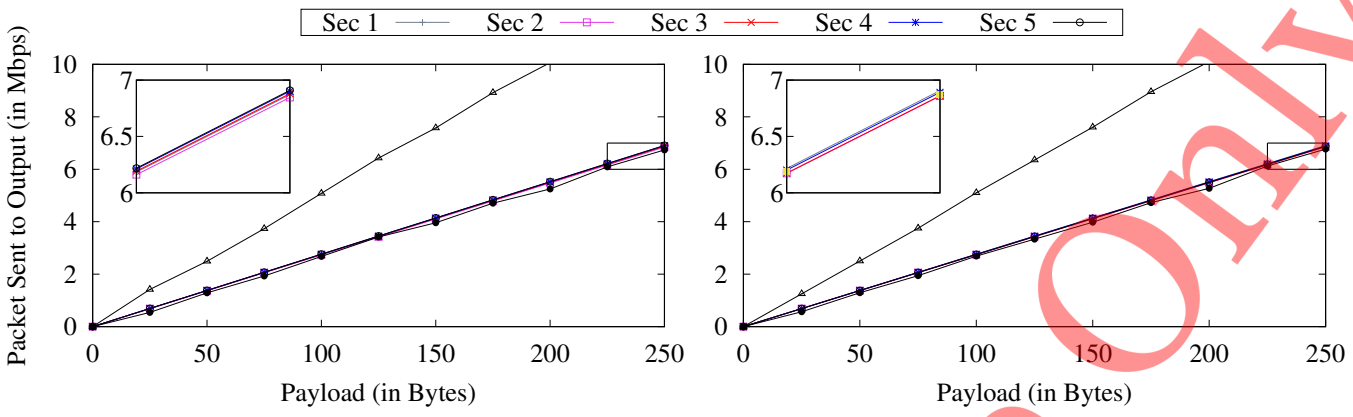
Based on aforementioned state-transition conditional probability, we calculate the probability of the packet to be at the aforementioned states. Based on Equation (1), the probability that the packet has to be at i^{th} position of the queue, B_i, where $0 \leq i \leq Q$, i.e., $P(B_i)$, at least once is defined as follows:

$$P(B_i) = \sum_{j=i}^{Q} \frac{p'}{Q+1} = p'\left(1 - \frac{i}{Q+1}\right), \quad 0 \leq i \leq Q \quad (10)$$

where $P(C)$ defines the probability of packet to be at SDN controller. We consider that $P(C) = 1$. Based on Equation (10), the probability $P(F_0)$ of the packet to be processed at the ingress flow-table F_0 is calculated as $P(F_0) = P(F_0|B_0)P(B_0) = p'$. The packet reaches the ingress flow-table F_1, if and only if the packet gets forwarded by the ingress flow-table F_0. Hence, from Equation (4), the probability $P(F_1)$ of packet being in the ingress flow-table F_1 is defined as $P(F_1) = P(F_1|F_0)P(F_0) = \frac{p'}{3n}$. Additionally, from Equation (4), the probability $P(F_j)$ of the packet getting processed at the ingress flow-table F_j, where $1 < j \leq N$, is calculated as follows:

$$P(F_j) = \sum_{j'=0}^{j-1} P(F_j|F_{j'})P(F_{j'}) = \frac{p'}{3N}\prod_{j'=1}^{j-1}[1 + \frac{1}{3(N-j')}] \quad (11)$$

From Figure (1), we observe that the packet can reach to egress flow-table F_e from any ingress flow-table F_j, where $0 \leq j \leq N$. Hence, based on Equations (4), (5), and (11), the probability $P(F_e)$ of the packet to be processed at the egress flow-table F_e is defined as follows:



(a) Arrival Rate = 199147 pps (b) Arrival Rate = 200633 pps

Fig. 2: Sent to Output Rate of an OpenFlow Switch

$$P(F_e) = \sum_{j=0}^{N} P(F_e|F_j)P(F_j) \tag{12}$$

Additionally, we observe that the packet can only reach the egress flow-table F_{e+1} from the egress flow-table F_e. Hence, using Equations (6) and (12), we get the probability $P(F_{e+1})$ of the packet to be processed at egress flow-table F_{e+1} is as follows:

$$P(F_{e+1}) = P(F_{e+1}|F_e)P(F_e) = \left(\frac{2+p}{6M}\right) P(F_e) \tag{13}$$

Using Equations (6), (12), and (13), we define the probability $P(F_{e+k})$ of the packet being processed at egress flow F_{e+k}, where $1 < k \leq M$, as follows:

$$P(F_{e+k}) = \frac{(2+p)}{6M} P(F_e) \prod_{k'=1}^{k-1} \left[1 + \frac{2+p}{6(M-k')}\right] \tag{14}$$

1) Output Action Probability: As shown in Figure 1, the packet reaches the output action state from either ingress flow-table F_j, where $0 \leq j \leq N$, or egress flow-table F_{e+k}, where $0 \leq k \leq M$. Hence, the probability $P(O)$ of the packet being in the output action state depends on Equations (6), (7), and (11)–(14). We define $P(O)$ as follows:

$$P(O) = \sum_{j=0}^{N} P(O|F_j)P(F_j) + \sum_{k=0}^{M} P(O|F_{e+k})P(F_{e+k}) \tag{15}$$

2) Packet Drop Probability: From Figure 1, we observe that the packet reaches the packet drop state D from either ingress flow-table F_j, where $0 \leq j \leq N$, or egress flow-table F_{e+k}, where $0 \leq k \leq M$. Hence, the probability $P(D)$ of the packet getting dropped depends on Equations (6), (8), and (11)–(14). We define $P(D)$ as follows:

$$P(D) = \sum_{j=0}^{N} P(D|F_j)P(F_j) + \sum_{k=e}^{e+M} P(D|F_k)P(F_k) \tag{16}$$

3) Send to Controller Probability: From Figure 1, we get that the packet reaches the controller from either an ingress flow-table F_j, where $0 \leq j \leq N$, or an egress flow-table F_{e+k}, where $0 \leq k \leq M$. Hence, the probability of the packet getting forwarded to the controller depends on Equations (6), (9), and (11)–(14). We define $P(C)$ as follows:

$$P(C) = \sum_{j=0}^{N} P(C|F_j)P(F_j) + \sum_{k=0}^{M} P(C|F_{e+k})P(F_{e+k}) \tag{17}$$

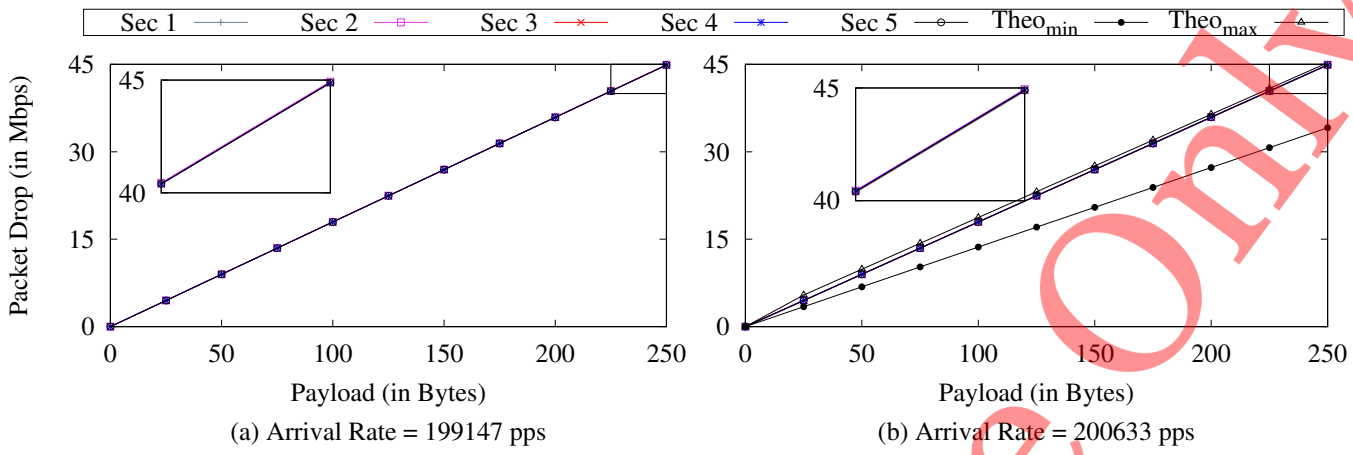
which is same as $P(D)$. Moreover, using *Stirling's approximation formula*, we evaluate the upper and lower bound for $P(F_e)$, $P(F_{e+k})$, $P(O)$, and $P(D)$.

IV. Performance Analysis

In this Section, using AMOPE, we analyze the performance of packet flow through an OpenFlow switch in SDN. We evaluate the performance of AMOPE based on the parameters mentioned in Section IV-B. We have simulated the proposed model in the MATLAB simulation platform. For simplicity, we consider that the number of OpenFlow switch in SDN is two, i.e., $p' = \frac{1}{2}$, where p' is the probability of the packet getting forwarded to the concerned switch. Additionally, we consider that if a packet is forwarded to the controller, it eventually, is queued in an OpenFlow switch.

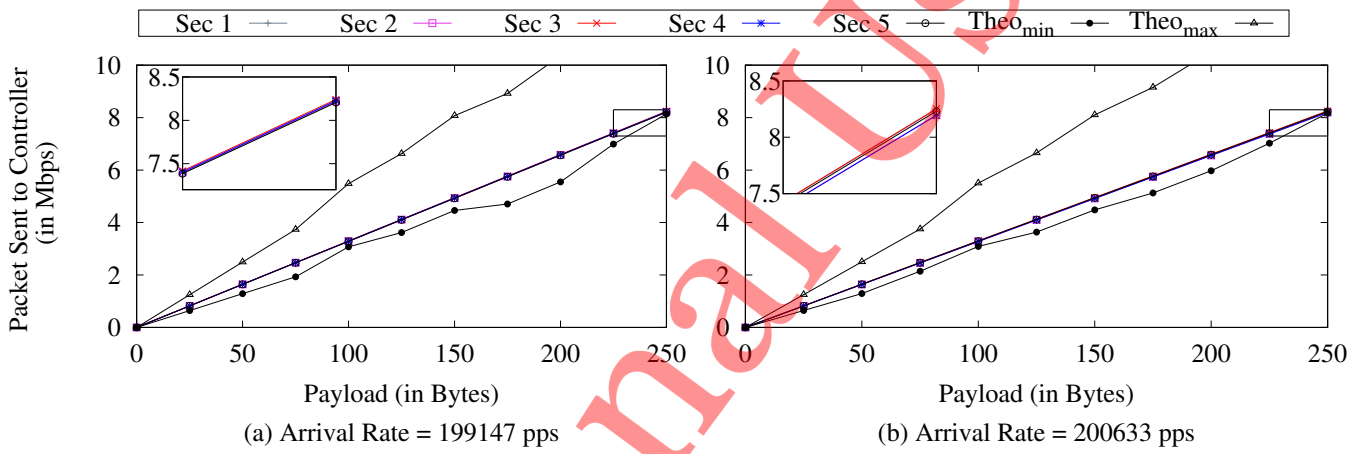
A. Simulation Parameters

In AMOPE, simulations are performed for the OpenFlow switch in SDN with a single controller and two OpenFlow switches. We consider that the packet arrival rate and the packet service rate per OpenFlow switch are approximately 0.2 million packets per second (mpps) [5] and 0.03 mpps [23], respectively. We consider different simulation parameters, as shown in Table I. The simulation time is 5 sec, queue size per OpenFlow switch is 0.73 million packets [5]. We consider that there are 10 number of ingress flow-tables and either zero or 10 number of egress flow-tables, as shown in Table I.



(a) Arrival Rate = 199147 pps (b) Arrival Rate = 200633 pps

Fig. 3: Packet Drop Rate of an OpenFlow Switch



(a) Arrival Rate = 199147 pps (b) Arrival Rate = 200633 pps

Fig. 4: Send to Controller Rate

TABLE I: Simulation parameters

Parameter	Value
Number of OpenFlow switch	2
Packet arrival rate per switch	0.199147, 0.199731, 0.200633 mpps
Packet service rate per switch	0.03 mpps [23]
Queue size per switch	0.73 million packets [5]
Flow table lookup time	33.33333 μsec [23]
Number of ingress tables	10
Number of egress tables	$\{0, 10\}$

B. Performance Metrics

We evaluated the performance of the OpenFlow switch based on the Markov chain-based analytical model with different packet arrival rates 0.199147, 0.199731, 0.200633 million packets per second (mpps), while considering the following parameters:

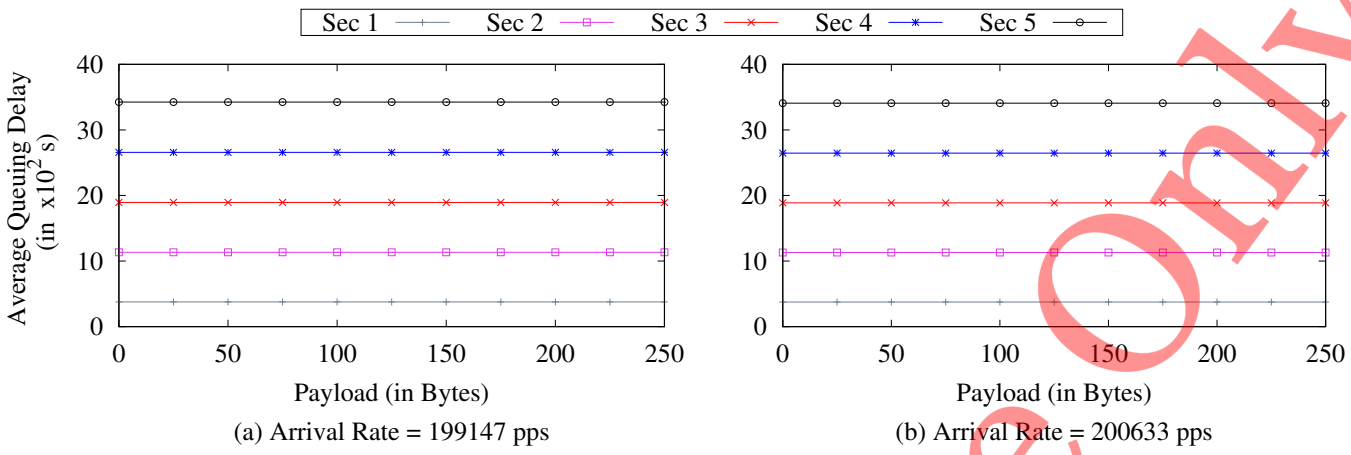
Throughput: We consider that the throughput of an OpenFlow switch is defined as the number of packets processed, i.e., reaches the output action state. A packet can reach the output action state from any ingress or any egress flow-tables.

Number of Packets Dropped: A packet can be dropped due to following reasons — there is no table-miss entry for ingress and egress flow-table or the table-miss entry is to drop the packet, the output action is not defined for matched entry at ingress or egress flow-table, and the action specified by the table-miss flow entry is drop.

Number of Packets Sent to Controller: A packet is sent to the controller if the action mentioned in the table-miss entry is to forward a packet to the controller. The packets, which are forwarded to the controller from the OpenFlow switches, are considered to be queued again in one of the available OpenFlow switches.

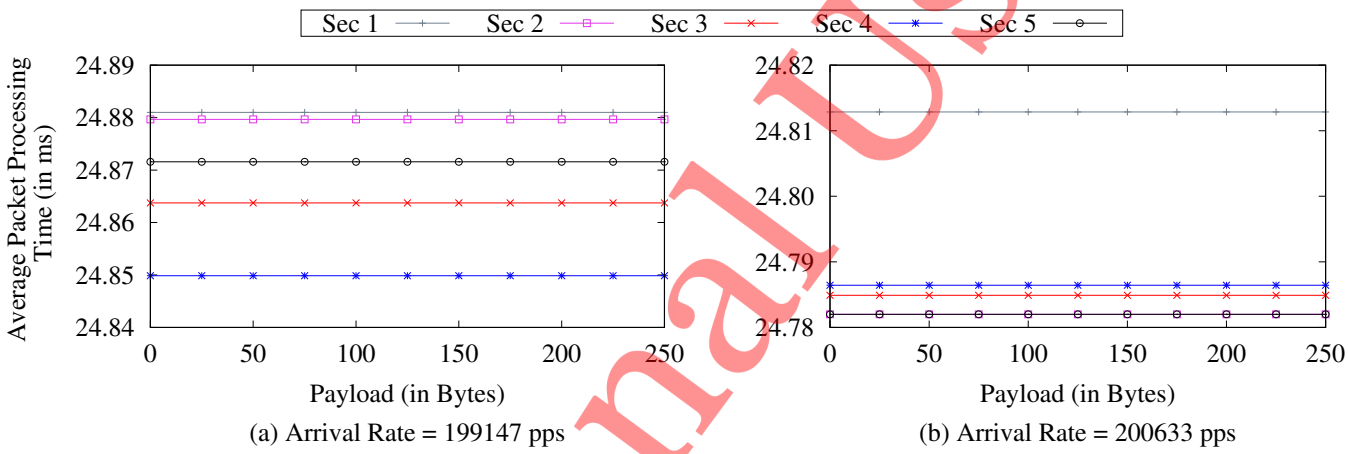
Average Queuing Packet Delay: We calculate the average queuing packet delay as the duration between time stamp when a packet enters into OpenFlow switch, and the time stamp when the packet enters through ingress port for processing.

Packet Processing Time: We consider the packet process time is the duration between the time stamp when a packet enters to ingress flow-table F_0 and the time stamp when the packet gets out of the switch.



(a) Arrival Rate = 199147 pps (b) Arrival Rate = 200633 pps

Fig. 5: Average Queuing Delay



(a) Arrival Rate = 199147 pps (b) Arrival Rate = 200633 pps

Fig. 6: Average Packet Processing Delay

C. Result and Discussion

In AMOPE, for simulation, we generated random numbers from the *Poisson distribution* with the mean packet arrival rate 0.2 mpps. Additionally, we considered randomness, while taking a decision on table-hit and table-miss. If there is table-miss flow entry, and action mentioned for table-miss flow entry is to forward to the controller, the packets get queued again in the OpenFlow switch buffer.

From Figure 2, we observed that approximately 60% of the arrived number of packets are sent for output action. In Figure 2, the throughput of an OpenFlow switch increases with the increase in payload size. Additionally, we get that the throughput in each second is almost similar. Hence, we conclude that the throughput of an OpenFlow switch depends on the payload as well as on the number of matched packets. On the other hand, Figure 3 shows that almost 9% of the packets are dropped, as there is either no table-miss flow entry for ingress and egress of flow-tables, the action mentioned in the table-miss flow entry is packet drop, or any output action is not mentioned in the matched entry. Here, based on Figure 3, we argue that the packet drop (in $Mbps$) increases with the increase in payload. Additionally, we conclude that OpenFlow considers each packet as an individual entity, and process separately. Additionally, from Figure 4, we yield that the approximately 31% of the arrived packets are sent to the controller, and sent back to OpenFlow switch queue again. From Figures 2, 3, and 4, we observed that the simulated results lie within the theoretical minimum and maximum values obtained using the proposed analytical model of an OpenFlow switch.

Figures 5 and 6 depict two types of delay of the OpenFlow switch such as queuing and processing delay. From Figure 5, we observed that the average queuing delay is much higher compared to processing delay. Hence, we conclude that the packet delay at OpenFlow switch increases mostly due to the packet queuing delay. On the other hand, from Figure 6, we observed that the average packet processing time is almost similar for each time instant. For each time instant, the packet processing delay is in the range $[33.3333\ \mu sec - 0.025\ sec]$. We get approximately 0.02 sec processing delay, in case of the packet has to go through for match entries for each ingress and egress flow-tables. In Figure 6, we observed that the average processing delay varies randomly for different arrival rate, as the packet processing delay solely depends on the number

of flow-tables the packet has to go through for finding a match. Additionally, from Figures 5 and 6, we yield that delay factor are almost linear with the variation of payload, as the processing time in an OpenFlow switch depends on the header size of the packet, i.e., the matched field entries, and does not depends on the payload.

Hence, we argue that the packet delay can be improved, while using an efficient queuing algorithm for an OpenFlow switch. On the other hand, the packet drop rate is too high for an OpenFlow switch due to limitations of TCAM memory size, and the mismatch of rules. Hence, we suggest that the packet drop rate can be improved, while using TCAM memory, efficiently, and using a proper rule placement mechanism.

V. Conclusion

In this paper, we analyzed the performance of packet flow through an OpenFlow switch in SDNs and proposed an analytical model, named AMOPE, to define the probabilistic bounds of the performance metrics of the OpenFlow switch. We modeled the packet flow steps in an OpenFlow switch using Markov chain, and calculated theoretical probabilities of the packet to be any state. Additionally, we have calculated the probabilities of a packet being at output action state, packet getting dropped, and packet getting forwarded to the controller, theoretically. We also verified the theoretical findings using the MATLAB simulation platform. We infer that in an OpenFlow switch, the total delay is high due to high delay at the queue of the OpenFlow switch. On the other hand, in an OpenFlow switch, a high number of packets get dropped due to either not having table-miss flow entry, or output action not being specified.

Future extension of this work is to propose an efficient queuing scheme, so that queuing delay of packet flow gets reduced significantly. In addition, this work can be extended to understand how packet drop rate can be reduced while using the available TCAM memory in an OpenFlow switch.

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," in *Proc. of IEEE*, vol. 103, no. 1, January 2015, pp. 14–76.

[2] OpenFlow. (2014, Dec.) OpenFlow Switch Specification Version 1.5.0. Open Networking Foundation.

[3] N. P. Katta, J. Rexford, and D. Walker, "Incremental Consistent Updates," in *Proc. of ACM SIGCOMM Works.* New York, NY, USA: ACM, 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. on Net.*, vol. 20, no. 2, pp. 488–500, April 2012.

[5] 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. of IEEE GLOBECOM*, December 2013, pp. 1397–1402.

[6] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and Performance Evaluation of an OpenFlow Architecture," in *Proc. of Int. Tele. Cong.*, 2011, pp. 1–7.

[7] U. Javed, A. Iqbal, S. Saleh, S. A. Haider, and M. U. Ilyas, "A stochastic model for transit latency in OpenFlow SDNs," *Comp. Net.*, vol. 113, pp. 218 – 229, 2017.

[8] R. Trestian, K. Katrinis, and G. Muntean, "OFLoad: An OpenFlow-Based Dynamic Load Balancing Strategy for Datacenter Networks," *IEEE Trans. on Net. Serv. Man.*, vol. 14, no. 4, pp. 792–803, Dec 2017.

[9] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for Network Update," in *Proc. of ACM SIGCOMM*, New York, NY, USA, 2012, pp. 323–334.

[10] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "OpenFlow Switching: Data Plane Performance," in *Proc. of IEEE Int. Conf. on Comm.*, May 2010, pp. 1–5.

[11] A. Rego, L. Garcia, S. Sendra, and J. Lloret, "Software Defined Networks for Traffic Management in Emergency Situations," in *Proc. of the 5^{th} Int. Conf. on SDS*, Apr. 2018, pp. 45–51.

[12] A. Rego, A. Canovas, J. M. Jimnez, and J. Lloret, "An Intelligent System for Video Surveillance in IoT Environments," *IEEE Access*, vol. 6, pp. 31 580–31 598, 2018.

[13] P. Kampanakis, H. Perros, and T. Beyene, "SDN-based solutions for Moving Target Defense network protection," in *Proc. of IEEE Int. Sym. on a World of Wireless, Mobile and Mult. Net.*, June 2014, pp. 1–6.

[14] K. Salah, K. Elbadawi, and R. Boutaba, "Performance Modeling and Analysis of Network Firewalls," *IEEE Trans. on Net. and Serv. Man.*, vol. 9, no. 1, pp. 12–21, March 2012.

[15] S. Misra, A. Mondal, and S. Khajjayam, "Dynamic big-data broadcast in fat-tree data center networks with mobile iot devices," *IEEE Systems Journal*, pp. 1–8, 2019.

[16] D. L. Eager and K. C. Sevcik, "Performance Bound Hierarchies for Queueing Networks," *ACM Transactions on Computer Systems*, vol. 1, no. 2, pp. 99–115, May 1983.

[17] G. Faraci and G. Schembra, "An Analytical Model to Design and Manage a Green SDN/NFV CPE Node," *IEEE Trans. on Net. and Ser. Man.*, vol. 12, no. 3, pp. 435–450, September 2015.

[18] 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. on Net. and Ser. Man.*, vol. 14, no. 3, pp. 603–615, September 2017.

[19] G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," *IEEE J. on Sel. Areas in Comm.*, vol. 18, no. 3, pp. 535–547, March 2000.

[20] A. Markov, "Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain," in *Dynamic Probabilistic Systems (Volume I: Markov Models).* John Wiley & Sons, Inc., 1971, pp. 552–577.

[21] M. A. Marsan, G. Conte, and G. Balbo, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems," *ACM Trans. on Comp. Sys,*, vol. 2, no. 2, pp. 93–122, May 1984.

[22] J. Suzuki, "A Markov Chain Analysis on Simple Genetic Algorithms," *IEEE Trans. on Syst., Man, and Cyb.*, vol. 25, no. 4, pp. 655–659, April 1995.

[23] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore, "OFLOPS: An Open Framework for OpenFlow Switch Evaluation," in *Proc. of Int. Conf. on Pas. and Act. Net. Meas.* Springer, 2012, pp. 85–95.