Delay Analysis of New-Flow Setup Time in Software Defined Networks

Amer AlGhadhban and Basem Shihada
Computer, Electrical, and Mathematical Sciences and Engineering (CEMSE) Division
KAUST, Saudi Arabia
{amer.alghadhban, basem.shihada}@kaust.edu.sa

***Abstract*—Software Defined Networking (SDN) provides network engineers with a high-level of abstraction to manage network traffic and control the associated network resources. Unfortunately, the data-plane devices communicate with the controller for every new flow, which adds an extra overhead and causes excessive delays. Such communication relies on the probability called matching probability. In this work, we propose a mathematical model for the SDN flow-setup process with the consideration of all factors that contribute into the matching probability, such as proactive/reactive flow setup modes. Finally, we attempt at deriving the overall system capacity and blocking probability.**

***Keywords*—Performance analysis; Software defined network; OpenFlow system capacity; Proactive and reactive flow modes.**

I. Introduction

OpenFlow is a common SDN protocol used for the south-bound communication between the control-plane and the data-plane devices. The data plane in an OpenFlow switch is defined by an action-based flow table, where each flow-entry consist of match-fields with corresponding actions, such as modify and drop. The match-fields consist of 12-tuples including source/destination MAC and IP addresses and an expiration-time (typically 60 seconds) [1]. When a data-plane device receives an unknown flow, it sends a flow request (i.e., `packet_in` message) to the control-plane device, defined as the controller, looking for the right action to handle the packets of the unknown flow. Practically, the controller is programmed with a routing protocol as well as network and security policies, such as rate-limiting and access control. The controller responds with a flow-entry contains the required information with the right action(s). Finally, the OpenFlow protocol installs the received flow-entry into the switch's flow-table.

From this procedure, we observe that the flow setup passes through several sequential services that are controlled by a probability of having a match in the flow-table. These services begin with the data-plane device, the communication channel, and the control-plane device, respectively. In general, the unknown flow is either matched by a proactively installed flow-entry or a reactively installed flow-entry. The reactive flow-entry setup is the process of installing/modifying a flow-entry in an on-line manner as a positive reaction to a network incident, e.g., rerouting a large flow from a heavy loaded path to a less loaded one. Whereas the proactive flow-entry setup is the process of installing a flow-entry in an off-line manner, such as installing the entries of an access-control list (ACL) or static routing before production or during non-peak times. We evaluate the implications of flow setup process on different sizes of network flows. We found that at low load, the examined throughput of large TCP flows dropped by a about 40% and the delay of round-trip-time flows was increased by a factor of $3\times$. As expected, the flow setup delays severely affect small flows.

In order to improve the flow-setup process, researchers proposed three main approaches. First, enhancing the controller response time by implementing a distributed linear or hierarchical control-plane system [2] [3] [4] [5]. Second, improving the response time of data-plane devices by fabricating most of the data-plane functions into hardware chips or speed-up the flow-table lookup time [6] [7] [8] [9]. Finally, the flow management approaches are not aimed at entirely avoiding the flow setup process. It is rather aimed at minimizing the interactions with the SDN controller through the adoption of flow based technique [10] [11] [12].

Thus, in this work, we aim at studying and analyzing the possible factors that contribute to the third approach, such as reactive/proactive flow setup modes, with different service time distributions. In the reactive flow setup, the arrival of a new flow and the expiration-time of a flow-entry have shown a hazy harmony. The arrival-rate of new flows and flow-entry expiration-time are functioning on a different time-scale. The flow-entry is added upon the arrival of a new-flow and removed when it expires. For example, the flow in data center network arrives in milliseconds time-scale, while the typical flow-entry expiration-time in OpenFlow switches is 60 seconds. Such difference in time-scale will lead into a non-steady-state condition when it is modeled as it is. In this work, we analyzed the reactive flow setup by using total probability theorem and Discrete-Time Markov Chain (DTMC) model.

After this introduction, the motivation and literature review are discussed. The problem formulation of the flow setup delay and where flow management techniques contribute in are illustrated in Sec. III. The mathematical description of system model including delay analysis of flow setup time, system capacity and blocking probability are presented in Sec. IV. The performance results are described in Sec. V. Finally, conclusions are drawn in Sec. VI.

II. Motivation

In this section, we investigate the extra overheads caused by OpenFlow protocol on data-plane and control-plane devices. We build our investigation using Mininet emulation [13] supported by Openvswitch 2.2.0 [14] and POX controller [15].

The Mininet was installed on top of Intel Xeon CPU X5550 2.67GHz 8 cores with 40GB memory. The POX controller was installed on a dedicated virtual machine. We implement five hosts as traffic sources and one server as a destination. Also, the controller is programmed to respond to reactive flow requests with the appropriate flow-entries and to configure the switches with the needed flow-entries for the proactive mode testing.

In this experiment, we measure the impact of flow-setup on TCP throughput and on the round trip time. The TCP throughput was measured by using Iperf; a commonly used tool for throughput measurement and we used a normal ping to measure the round trip time. In the testing of TCP throughput, we use multiple values for the flow duration; 5 and 10 seconds. Moreover, the TCP throughput of the communicating hosts is measured when there is an external load on the controller which mimic a more realistic scenario. The load on the controller is generated by using Cbench [16]. For the sake of accurate investigation, we coded the Mininet to send the TCP flows and ping messages according to the exponential distribution. Also, we use two different flow-entry expiration-times; 5 and 55 seconds, in the testing of both the reactive and proactive modes, as displays in Fig. 1. However, the produced delay of round-trip traffic in proactive mode is very small compared to reactive mode results, due to this we removed the proactive result in Fig. 1. The results are the average of 30 runs and the RTT values were measured from the five sources.

The impact of the flow-setup process on TCP throughput is illustrated in Fig. 2. The results show that the flow-setup decreases the TCP throughput by about 40%. The experiment results, as displays in Fig. 1, demonstrate how the flow-setup seriously contribute to the delay of round-trip time messages. This impact is due to the short life time of the round trip compared to the flow-setup time. This implies that the overhead of the flow-setup will severely increase the flow completion time of delay sensitive flows, defined in the literature as mice-flows. It is worth noting that the mice flow represents the majority of data center flows [10] [17]. Also, the results demonstrate that when the flow-entry expiration-time increases the round trip time decreases, as shown in the figure. When we set the expiration-time to 55 seconds, the round-trip time dramatically decreased. In literature, there are two

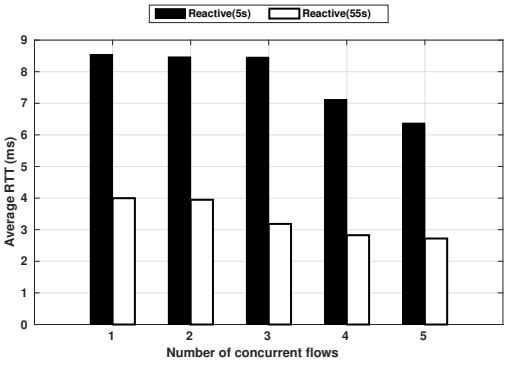

Fig. 1: The effect of flow-setup on round-trip-time.

kinds of studies investigated the performance figures of SDN

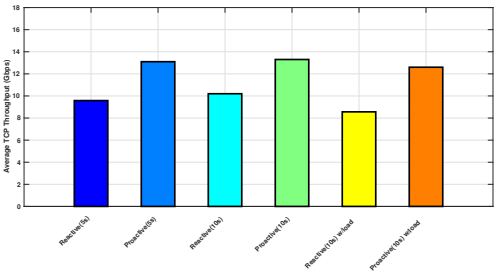

Fig. 2: The effect of flow-setup on TCP Throughput.

framework and OpenFlow protocol. Either using simulation supported by practical experiments, such as the work of DevoFlow [10] and Oflops [18] or mathematical model, such as the models proposed by [19] [20] [21] [22]. In the latter approach, the researchers exploit different mathematical modeling methods to understand the characteristics of OpenFlow protocol. The authors in [19] model the OpenFlow protocol by using M/M/1-S feedback-oriented queuing method where both of the data-plane and control-plane are represented by Markovian servers. They evaluated the performance figures of flow-setup delay in case the controller is involved in flow-setup and in case of otherwise. To obtain an accurate mathematical model of OpenFlow protocol, the authors of [20] utilized the network calculus theory to model the communication between OpenFlow data-plane and control-plane devices. The introduced model is used in their investigation of the performance of both OpenFlow planes. In [21], the authors used Jackson network with additional modifications to model the interaction between OpenFlow data-plane and control plane devices. Their model can be extended to represent more than one data-plane device. Finally, a mathematical model of the packet forwarding of OpenFlow switches as well as the processing of packet-in messages in the OpenFlow controller is introduced by Xiong et al. [22]. Since the flow management techniques demonstrated its significant contribution in accelerating the flow-setup process [10] [11] [12], we are the first to study its impact on the performance of OpenFlow communications by using MATLAB simulation supported by mathematical analysis.

III. Problem Formulation

In OpenFlow, a new flow has to pass through a sequence of services started from the data-plane device d_{swf}, the communication channel, d_{ch}, the control-plane device, d_{cn}, then back to the switch. In the flow-setup process, the switch is visited twice; when a new flow arrives and when the flow-entry installed into the flow-table. In the sake of accuracy, we use different service times for the two visits; the first visit is, d_{sws}, and the second visit is d_{swf}. The service delay, D_{fs}, for the flow setup is given by:

$$D_{fs} = d_{swf} + d_{sws} + d_{ch} + d_{cn} \quad (1)$$

Since, there is a probability of having a match, defined as matching probability Pr_{match}, with a flow-entry installed to handle previous flows, the flow service delay is given by,

$$D_{fs} = d_{swf} + (1 - Pr_{match}) \cdot [d_{sws} + d_{ch} + d_{cn}] \quad (2)$$

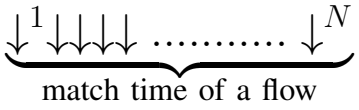

Fig. 3: Reactive flow-setup mode

To understand the OpenFlow system and the flow management techniques, we present a summary of the different types of flow-setup modes in the following subsections.

A. Reactive Flow-Setup Mode

Typically, in the reactive flow-setup, the data-plane device sends a flow request to the controller per the arrival of a new flow. Then, the controller installs a flow entry in response. The newly installed flow-entry could be matched by any subsequent flows. However, the likelihood of this match to happen depends on three factors. First, the flow-entry expiration-time, τ. Second, the flow arrival-rate, λ. Finally, the number of flow-entry, S, in the data-plane's flow-table. When a recently installed flow-entry is prolonged to last for τ_{max} expiration-time, the likelihood of a flow from subsequent flows to match this flow-entry is increased. Also, when the minimum bound of flow inter-arrival time is less than τ_{max}, the recently installed flow-entries will expire before the arrival of a new flow. For instance, when the expiration-time is τ_d and the minimum bound of inter-arrival time is e_m, where $e_m > \tau_d$, the expiration-time τ_d will elapse before the arrival of a new flow from the same class. Finally, when a flow arrives at a data-plane device that has an empty flow-table, definitely the matching probability is zero. Thus, the matching probability increases with the increase in any of these factors. The impact of these factors are proved analytically in this section, and their empirical results are displayed in Sec. V.

Modeling the probability in the reactive mode needs to obtain a mathematical relationship between the subscribed three factors. We start by understanding the relationship between the arrival-rate and expiration-time. Assuming $\overline{N}$ is the average number of flows arrives at an interval of time τ_i, as shown in Fig. 3, where flow#N is the first flow arrived during τ_i. If its flow-entry expiration-time lasts until an arrival of a new flow, flow#1, that has the same matching fields, then it will be matched by flow#N's entry and bypass the extra overhead of the flow-setup process. According to Little's theorem, the average number of flows during any interval of time relies on the flow arrival-rate as well as flow-entry expiration-time, where $\overline{N} = \lambda \cdot \tau_i$.

On the other hand, when a new flow doesn't match with one of the existing flow-entries, S, of a data-plane device, where $S \in \{0, 1, 2, 3..., K\}$, a new flow-entry is installed and accordingly the value of S increases. As a result, the matching probability in the reactive mode when the flow-table has S number of flow-entries is given by,

$$Pr_{react} = \frac{S}{K} \quad where, \quad 0 \leq S \leq K \tag{3}$$

The likelihood of having a new flow-entry is dependent on the matching probability, where the matching probability is increasing with the increase in S. As discussed above, the value of S in reactive mode changes according to the arrival-rate as well as the expiration-time. For an accurate representation of (3), it is necessary to include the probability $P_{S(\tau)}$ of having a certain number of flow-entries S.

$$Pr_{react} = \frac{S}{K} \cdot P_{S(\tau)} \tag{4}$$

The challenge is how to model this probability and how to express its relation to other factors? It is obvious that the flow-entry in the reactive mode is incremented for every mismatch. Additionally, the flow-entry lasts for a particular time τ; then it is removed from the flow-table. The transition process of adding a new flow-entry per a mismatch and removing one per expiration has a characteristic similar to a birth-and-death process. The problem is that the arrival-rate is working in a millisecond timescale, while the flow-entry removal is working in a multiple seconds timescale. Which means that the arrival-rate is much faster than the departure rate, where this leads into a non-steady-state system.

In order to solve the aforementioned challenge, we use a discrete-time Markov chain model as shown in Fig. 4 to represent the mathematical relationship between the three factors and to get the $P_{S(\tau)}$ probability. During a small-time instant τ_0 where only one event from the following probabilities can appear. First, the probability of a new-flow matching one of the current flow-entries $p \cdot \frac{S}{K}$. Second, the probability of a new-flow doesn't matching any of the current flow-entries, i.e., a mismatch event, $(1 - \frac{S}{K})p$. Third, the probability of a flow-entry expires is $(1 - p)$, where p is a Geometric probability which contains the probability p_{τ_0} of whether having an arrival or otherwise during τ_0. The probability p is repeated all over the way until all the time instant, τ_0, of τ, is examined. In the same context, p_{τ_0} is a Poisson distribution to consider the probability of having a single event during τ_0. The Geometric probability ensures that the event will appear during the whole period of τ and the Poisson probability is used to ensure of having a single event during the small instant of time τ_0. According to the DTMC model, the state probabilities are as the follows,

$$p = P[X \leq \tau/\tau 0] = \sum_{n=1}^{\frac{\tau}{\tau_0}} p_\tau \cdot (1 - p_\tau)^n \tag{5}$$

$$(1 - p) = P[X > \tau/\tau 0] \tag{6}$$

$$p_{\tau_0} = (\lambda \tau_0) \cdot exp\,(-\lambda \tau_0) \tag{7}$$

$$P_0 + P_1 + P_2 + P_3 + ... + P_K = 1 \tag{8}$$

$$P_1 = \frac{pP_0}{(1 - p)} \tag{9}$$

$$P_K = \frac{(1 - \frac{K-1}{K} \cdot p)P_{K-1} - p \cdot (1 - \frac{K-2}{K})P_{K-2}}{(1 - p)} \tag{10}$$

$$P_M = \frac{p \cdot (1 - \frac{K-1}{K})P_{K-1}}{(1 - \frac{K-1}{K} \cdot p)} \tag{11}$$

Finally, the Markov model contains the impact of arrival-rate and expiration-time within the state probability $P_{S(\tau)}$. Since the matching probability in reactive mode increases with the increase in the number of states, i.e., flow-entries, and from

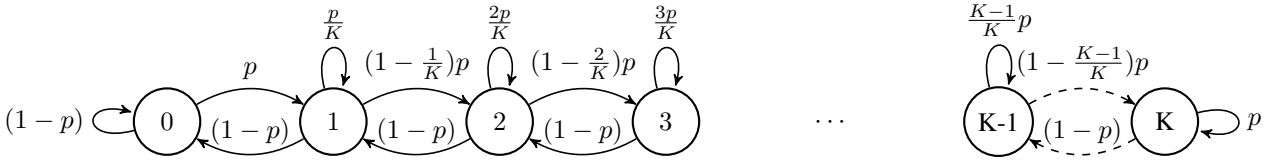

Fig. 4: DTMC state diagram of the arrival and departure of flow-entries into the flow-table

the total probability theory the reactive probability Pr_{react} will be,

$$Pr_{react} = \sum_{n=1}^{S} \frac{n}{K} \cdot P_{S(\tau)} \tag{12}$$

B. Proactive Flow-Setup Mode

Assuming K is the number of flow-entries that is required to match all classes of traffic to ensure the matching probability Pr_{match} is equal to 1. The controller, in network startup phase, installs several flow-entries in data-plane devices to match certain classes of traffic E where $E \subset K$, such as Address Resolution Protocol (ARP), and/or HTTP flows. When a flow from E arrives at a data-plane device, it will find a flow-entry to match its flow information. When a data-plane device has in its flow-table a flow-entry matches the information of a new flow, the data-plane doesn't need to communicate with the controller. Which means that the flow-setup and its overhead are neglected. The modeling of the matching probability in proactive mode, Pr_{proact}, depends on the share of the selected class among all existing classes of traffic. The traffic share of the selected class, C_i $i \in K$, is defined as how likely their flows are exchanging on the network of interest. For instance, since the share of HTTP traffic on the Internet is higher than other Internet traffic, when a flow is randomly picked from current Internet flows, is more likely to be HTTP flow. When flow-entries that handle all HTTP flows are proactively installed at a data-plane device, the probability of a new flow to be matched is almost equivalent to the share of HTTP flows among all the Internet flows [10].

$$Pr_{proact} = \frac{C_i}{\sum_{j=1}^{K} C_j} \forall j \in K \tag{13}$$

IV. Analysis

In this section, we present an analytical expression for the flow-setup delay experienced by all classes of the flows. We are interested in obtaining the mean waiting time $\overline{W}$ and its second moment $\overline{W^2}$ that will be experienced by all the flows at OpenFlow system. To achieve our goal, we first derived the mean results of the waiting time of all components in the OpenFlow system. The data-plane device, as well as the southbound channel, were modeled as $M/M/1$ with service times μ_1, and μ_2, respectively. To model the control-plane device, we use $M/G/1$, where this due to the diversity of services that can be provided by the controller. Since the summation of Poisson distributions is a Poisson distribution [23], the load on the controller from other data-plane devices were considered as γ which is a sum of λs that came from other data-plane devices. The mathematical analysis of control-plane used in this section considers standard derivation steps of $M/G/1$ model (see for instance [23]), and Table. I has the definitions of the main notations. However, the definitions of other notations were edited in the incoming paragraphs. The waiting time when the matching probability was considered is,

TABLE I: Summary of Important Notations

Symbol	Definition
Pr_{match}	The sum of both proactive and reactive matching probabilities.
K	The maximum number of flow-entries to have Pr_{match}=1 .
I	Flow index while in system
$P_{S(\tau)}$	Probability of having S flow-entries in the flow-table during τ.
λ_i	Poisson arrival rate of flow with index i
γ	The sum of λs arrive at the controller from other data-plane devices.
X_i	Service time random variable for flow i
R	Residual service time
W_{swf}	Waiting time in the switch for `packet_in` processing.
W_{sw^s}	Waiting time in the switch for the installation of a new flow-entry.
W_{cn}	Waiting time in the queue of the controller.
W_{ch}	Waiting time in the queue of the southbound channel.
μ_{swf}	The processing time of `packet_in` request in the switch.
μ_{ch}	The processing time of `packet_in` request in the channel.
μ_{cn}	The processing time of flow-setup request in the controller.
μ_{sw^s}	The processing time of flow-setup request in the switch.
T^f	Average response time of a flow for flow-setup process

$$\overline{W} = \overline{W}_{swf} + (1 - Pr_{match}) \left[\overline{W}_{cn} + \overline{W}_{ch} + \overline{W}_{sw^s}\right] \tag{14}$$

Since, the service time of the controller was considered as a general distribution, the average waiting time $\overline{W}_{cn}$ of a flow in the queue of the controller is given by,

$$\overline{W}_{cn} = \sum_{i=1}^{N} X_i + R \tag{15}$$

where X_i is the service time of packet i and R is the residual time. The mean residual service time $\overline{R}$ appearing in $\overline{W}_{cn}$ can be derived by the same kind of graphical $triangle\ trick$ as in the case of the (P-K) mean value. The first moment of this residual time can be written as,

$$\overline{R} = \frac{1}{2}\left[\rho \cdot \frac{\overline{X^2}}{\overline{X}}\right] \tag{16}$$

Making additional algebraic manipulations yield the desired second moment of waiting time as,

$$\overline{W^2} = \overline{N} \cdot \mathrm{Var(X)} + \left[\left(1 + \frac{\rho}{1-\rho}\right)\overline{\mathrm{R}}\right]^2 + \mathrm{Var(R)} \tag{17}$$

Where $Var(R) = \overline{R^2} - \overline{R}^2$. Also, (17) is procured by raising both sides of (15) to the 2[nd] power and taking the mean. Note

that the variables of $\overline{W^2}$ are all known except that we need to evaluate $\overline{R^2}$. Thereby, the law of total expectation, which states that E[Y] = E[E[Y|X]] is employed to obtain,

$$\overline{R^2} = \frac{1}{3}\left(\gamma \cdot \overline{X^3}\right) \tag{18}$$

A. Flow-Setup Delay

The average time a flow f spends in the OpenFlow system is given by,

$$\overline{T}^f = \overline{W}^f + \overline{X}^f \tag{19}$$

Where the average response time of the controller and subsequent services when the matching probability was considered,

$$\begin{aligned}\overline{T}^f =& 1/\left(\mu_{sw^f} - \lambda\right) + \left(1 - Pr_{match}\right) \cdot [\frac{\overline{R}}{(1-\rho_{cn})} + \overline{X}^f_{cn} \\ & 1/\left(\mu_{ch} - \lambda\right) + 1/\left(\mu_{sw^s} - \lambda\right)]\end{aligned} \tag{20}$$

The total response time distribution is,

$$\begin{aligned}f(t) =& T(\mu_1, \lambda)_{m/m/1} + \left(1 - Pr_{match}\right) \cdot [T(\mu_2, \lambda)_{m/m/1} \\ & + T(\mu_3, \lambda)_{m/m/1} + T(\mu_4, \gamma)_{m/m/1}].\end{aligned} \tag{21}$$

In the sake of accurate representation, we used different service time for each step in the flow-setup journey, where $\mu_1 < \mu_2 < \mu_3 < \mu_4$ and these are the service time of the switch, the controller, the southbound channel, and the processing overhead of flow-entry installation in a switch.

B. System Capacity

In a real network, the flow duration (i.e., the time since it starts till terminates) varies from certain service to another. For instance, the flows in data center network vary in size and duration, where the dominant flows last for less than a one second [17]. Thus, the flow-setup delay needs to be bounded by a particular time of quality of service. Otherwise, some of the new flows will wait in the setup queue for a time exceeds their duration. In this case, we need to model OpenFlow system when the queue size is bounded by some amount, defined here as Z. Accordingly, in this section, we try to understand the OpenFlow system capacity and what is the maximum number of flows under a certain response time quality of service $T - qos$ the system can handle.

$$Z^* = \underset{Z}{\operatorname{argmax}} \left\{ \sum_{i=1}^{Z} T_i^f \leq T_{qos} \right\}. \tag{22}$$

The system response time is a random variable as shown in (21), T_i, where $\{i = 1, 2, 3, ..., Z\}$ is the flows' response time index. When we use the exponential distribution as a service time of control-plane, the response-time will be,

$$\begin{aligned}T_i \approx & Exp(-\Lambda_1) + \left(1 - Pr_{match}\right) \cdot [Exp(-\Lambda_2) + \\ & Exp(-\Lambda_3) + Exp(-\Lambda_4)]\end{aligned} \tag{23}$$

The summation of the Z response times that resides in ahead of the last arrived flow in the queue will be, $\sum_{i=1}^{Z} T_i^f$. In order to get the system capacity, we need a closed formula for the distribution of T which is necessary to formulate the blocking probability. Since, it is known in the literature if χ_i where $\{i = 1, 2, 3, ..., L\}$ is i.i.d Exponential random variables with the constant parameter μ, the PDF of the summation is represented by Erlang distribution with L and μ parameters $E(L, \Lambda_i)$. However, here we have different Λs for each Exponential random variable, and we found the distribution of the total response time in (21) takes shape close to the exponential distribution with large L, where the Erlang and Gamma distributions tend to take the bell shape as the value of L increases. Therefore, the approximated distribution is Hypo-exponential $Hypo(\Lambda_1, ..., \Lambda_l)$,

$$\sum_{i=1}^{L} T_i^f \approx \sum_{i=1}^{L} \Lambda_i \cdot Exp(-\Lambda t) \left\{ \prod_{j=1, j\neq i}^{K} \frac{\Lambda_j}{\Lambda_j - \Lambda_i} \right\}. \tag{24}$$

The blocking probability is defined as, the flow-setup delay when it exceeds certain threshold value T_{Qos}, $\mathrm{P}[\mathrm{T} > \mathrm{t} = \mathrm{T_{Qos}}]$, which is necessary to understand the system capacity. The value of system response time is obtained from the response time distribution T explained above. In a particular case, the maximum number of flow-setup requests, L^*, that the system can handle before the T_{Qos} value is exceeded can be approximated to the following equation where $\overline{T}$ is the average response time,

$$Z^* \approx \frac{T_{Qos}}{\overline{T}} \tag{25}$$

In the evaluation section, we compared the Hypo distribution with the distribution of the system response time T when the Exponential distribution is used as the controller service time, μ_2.

V. Results and Discussion

We wrote a simulation model by using Matlab for SDN flow-setup process with the consideration of the two types of matching probabilities. The number of flows and the average arrival-rate in this evaluation are $K = 10^5$ and $\lambda = 1\frac{flow}{ms}$, respectively. Initially, as a proof of concept and to examine the validity and robustness of our simulation we compared the analytical and empirical results of a different version of $M/G/1$ by using different distributions as a service time, (Exponential, Gaussian and Erlang-2 distributions), the results are shown in Fig. 5.

The flow-setup delay when both matching probabilities (i.e., proactive and reactive) being used are shown in Fig. 6. From the figure, we can see how the results of the proposed mathematical model and the simulation are overlapped. This result demonstrates the accuracy of the proposed model. Also, we study how the matching probability is influencing the performance of both data-plane and control-plane service times. We found that when we increased the matching probability by $\times$ times the system performs much better compared to increasing the controller service time by the same amount. This means that the performance of the network is improved when the flow-setup overhead is avoided and a little enhancement in matching probability, like keep handling certain classes of flows in the switches, is better than increasing the controller service time. Moreover, the results prove that the data-plane device is the bottleneck in the flow-setup process. For instance, when we increased its service time by the same amount, the flow-setup delay decreased dramatically. The results are shown in Table. II. To investigate the validity of the ap-

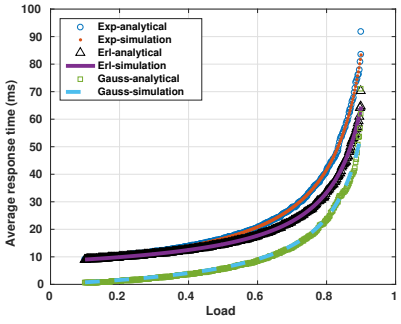

Fig. 5: Control-plane with different service time dist.

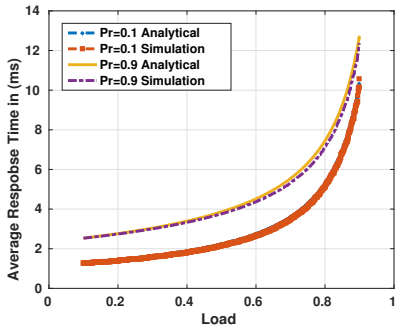

Fig. 6: The average response time of flow-setup delay.

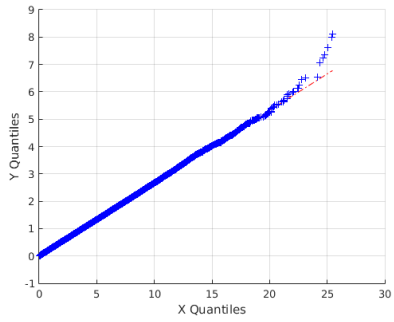

Fig. 7: The proposed model with Hypo-dist.

TABLE II:
Compare of increasing the service time of OpenFlow system with the Pr_{match}.

Pr_{match}	SW_{serv}	CN_{serv}	$\overline{T}$	↓ Percent(%)
0.45	1×	1×	4.614 ms	benchmark
0.45	1×	2×	4.3894 ms	-4.8695
0.45	2×	1×	1.9105 ms	-58.5940
0.9=2×	1×	1×	3.6218 ms	-21.5052

proximation of the response time, we used the QQ-plot of the proposed response time expression, i.e., (24) compared to hypo-exponential distributions, the results are shown in Fig. 7. We study the distribution density of the proposed model, where the PDFs of the system response time T, when Exponential, Gaussian, and Erlang-2 random variables were used as a service time of the controller, are shown in Fig. 8. We can see the three PDFs are almost overlapped.

We examine the effect of proactive probability alone on flow-setup delay. Since the proactive probability is based on the traffic classes, we used uniform distribution $\upsilon(0,1)$ to generate the proactive probability. Fig. 9 shows the response time with different values of proactive probability. Apparently, when the proactive probability increases, the response time decreases due to the avoidance of flow-setup delay.

For an accurate evaluation of reactive probability and to demonstrate the proposed DTMC-based model of reactive probability, we select different values of τ, λ, and M, where M is the number of flow-entries that is needed to get Pr_{match}=1. Also, we initialize the computation of reactive probability with P_0=$\frac{\lambda \cdot \tau}{M}$. Fig. 11 shows the proposed reactive flow setup model with different values of τ. From the figure, we observe that the matching probability increases proportionally to the increase in the value of τ. The value of M in these experiments is 1K. Also, we find that when we increase the arrival-rate λ the reactive probability increases as well. Fig. 10 displays the matching probability when we fixed the value of τ at 45 seconds and changed the value of λ. To understand the effect of the number of non-empty match fields M we fixed the value of τ at 45 seconds and the value of λ at 2.6 $\frac{flow}{ms}$, then we vary the value of M between 200 till 1000. We find

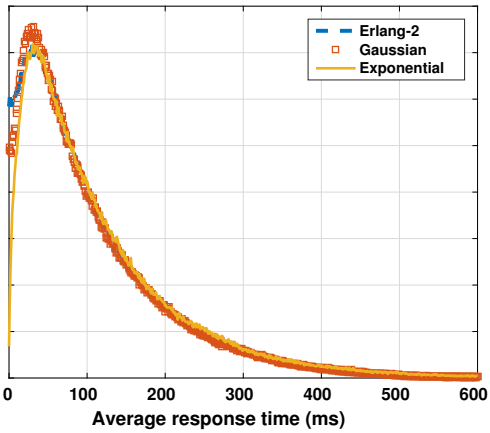

Fig. 8: PDF of the proposed model with different dist.

that the reactive probability is decreased with the increase in the value of M, as displayed in Fig. 12. Additionally, Fig. 10 displays the response time with reactive probability when we fixed the value of τ, also, at 45 seconds and changed the value of λ. The effect of λ values contribute not only to the reactive probability but also on the response time itself, and this is the reason why we get fast response time when we vary the λ's values. Practically, the minimum value of τ is more significant than the maximum value of λ since the value of λ is an r.v., unlike the value of τ which can be set by the network engineer.

VI. Conclusion

In this work, we study the impact of *matching probability* on the performance of OpenFlow. We found that increasing the share of flow management techniques is better than improving the performance of the controller. Also, we study the distribution of the system response time T, and we found that the distribution maintains its main characteristics even when we use different distributions for the controller service-time. Practically, the existing network systems have a limited capacity. Accordingly, we analytically study the system capacity and blocking probability, and in their empirical evaluation, we found that the distribution of T takes shape close to the exponential distribution with large values, unlike Erlang and Gamma distributions.

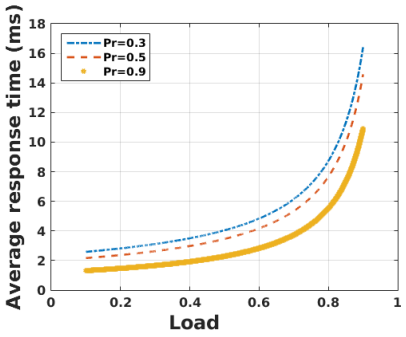

Fig. 9: Proactive probability and flow-setup delay.

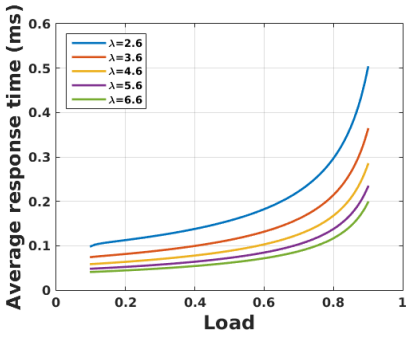

Fig. 10: Reactive probability with different values of λ.

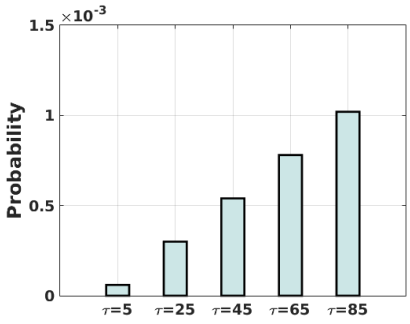

Fig. 11: Reactive probability with different values of τ.

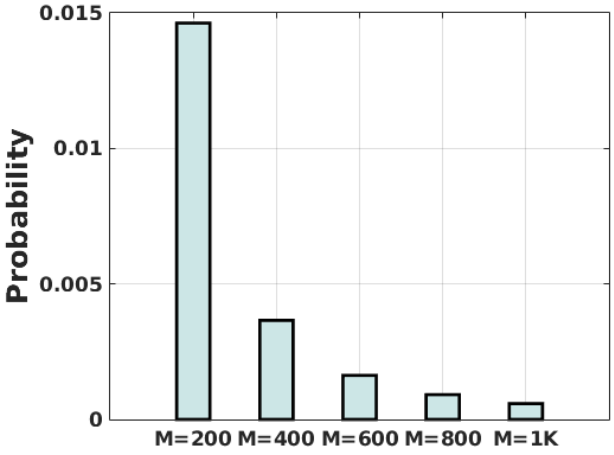

Fig. 12: Reactive probability with different values of M.

REFERENCES

[1] O. Manual. [Online]. Available: http://www.openvswitch.org/support/dist-docs/ovs-testcontroller.8.txt.

[2] A. A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Elasticon: An elastic distributed sdn controller," in *Proceedings of the Tenth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ser. ANCS '14, 2014, pp. 17–28.

[3] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12, 2012, pp. 19–24.

[4] T. Koponen et. al., "Onix: A distributed control platform for large-scale production networks." in *OSDI*, vol. 10, 2010, pp. 1–6.

[5] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, 2010, pp. 3–3.

[6] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 339–350, 2011.

[7] S. Zhouet et al., "A flexible and scalable high-performance openflow switch on heterogeneous soc platforms," in *Performance Computing and Communications Conference (IPCCC), 2014 IEEE International*, Dec 2014, pp. 1–8.

[8] P. Bosshart et al., "Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13, 2013, pp. 99–110.

[9] A. AlGhadhban and B. Shihada, "Energy efficient sdn commodity switch based practical flow forwarding method," in *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE, 2016, pp. 784–788.

[10] A. R. Curtis et al., "Devoflow: Scaling flow management for high-performance networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11, 2011, pp. 254–265.

[11] Z. Su, T. Wang, Y. Xia, and M. Hamdi, "Cheetahflow: Towards low latency software-defined network," in *2014 IEEE International Conference on Communications (ICC)*, 2014, pp. 3076–3081.

[12] M. Yu et al., "Scalable flow-based networking with difane," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 351–362, Aug. 2010.

[13] N. Handigol et al, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '12, 2012, pp. 253–264.

[14] B. Pfaff et al, "Extending networking into the virtualization layer," in *In: 8th ACM Workshop on Hot Topics inNetworks (HotNets-VIII).New YorkCity,NY(October 2009*, 2009.

[15] POX. [Online]. Available: http://www.noxrepo.org/pox/about-pox/.

[16] R. Sherwood and K. Yap, *Cbench: an openflow controller bench marker*. [Online]. Available: http://www.openflow.org/wk/index.php/Oflops.

[17] T. Benson et al, "Understanding data center traffic characteristics," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 92–99, Jan. 2010.

[18] C. Rotsos et al, "Oflops: An open framework for openflow switch evaluation," in *Proceedings of the 13th International Conference on Passive and Active Measurement*, ser. PAM'12, 2012, pp. 85–95.

[19] M. Jarschel et al., "Modeling and performance evaluation of an openflow architecture," in *Teletraffic Congress (ITC), 2011 23rd International*, 2011, pp. 1–7.

[20] S. Azodolmolky et al., "An analytical model for software defined networking: A network calculus-based approach," in *Global Communications Conference (GLOBECOM), 2013 IEEE*, 2013, pp. 1397–1402.

[21] M. Kashif et al., "On the modeling of openflow-based sdns: The single node case," *CoRR*, vol. abs/1411.4733, 2014. [Online]. Available: http://arxiv.org/abs/1411.4733

[22] 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.

[23] D. Bertsekas and R. Gallager, *Data Networks (2Nd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.