2019 International Conference on Computing, Networking and Communications (ICNC): Communications QoS and System Modeling

Queueing Analysis of Auxiliary-Connection-Enabled Switches for Software-Defined Networks

Chuanji Zhang, Hemin Yang, George F. Riley, and Douglas M. Blough
School of Electrical and Computer Engineering, Georgia Institute of Technology
{jenny.zhang, hyang350, riley, doug.blough}@ece.gatech.edu

***Abstract*— Software-Defined Networking (SDN) is a powerful technology to enable future network innovations, especially for data center networks. However, the limited capacity of the switch-to-controller link is a scalability issue for data center deployments. Auxiliary connections are introduced in the OpenFlow 1.3 Specification to help alleviate switch-to-controller link overflow and improve the scalability of SDN. In this paper, an analytical model is proposed to study the performance of auxiliary connections with OpenFlow switches. This model characterizes different interactions between the controller and the auxiliary-connection-enabled switches with limited buffer space. The model produces evaluation results in seconds for switches with realistic data center traffic loads, which would require days of compute time with network simulation. The analytical model is validated for lower traffic loads using ns-3 and an example in-depth analysis of switch performance is carried out with typical data center traffic loads to illustrate the utility of the approach. To the best of our knowledge, this is the first paper to study auxiliary connections for SDN analytically.**

I. Introduction

Software-Defined Networking (SDN) is an emerging network technology which enables flexible network management and faster network innovations. It has been a popular candidate for data center network since its emergence. In SDN, the controller configures the network behaviors by installing flow entries on the switches using southbound protocols. OpenFlow is the *de facto* southbound protocol of SDN and is envisioned as its enabler. However, due to the large traffic demand in the data center and the weak capacity of OpenFlow switches, the scalability issues arise. One of the issues is the limited capacity of the switch-to-controller link. An edge switch in an enterprise data center can see 100k new flows per second [1], which corresponds to a large *packet_in* traffic load on the switch-to-controller link. The demand will get much higher on the core switches, in commercial cloud data center, and as the usage of DCN is increasing dramatically recently. Besides, the collection of statistic and network status is also conducted via the switch-to-controller link. Such communications contribute to the traffic load on the switch-to-controller link by a non-negligible fraction [2]. Moreover, in-band SDN is proposed [3] and reinforces the need to have sufficient switch-to-controller link bandwidth. In order to address this challenge, auxiliary connections were introduced from OpenFlow 1.3 [4] to exploit the parallelism of switch implementation. The auxiliary connection is already enabled by many open-source SDN controllers such as OpenDaylight. Besides, some academic works also propose auxiliary connections are beneficial [5]. However, the auxiliary connection is not fully studied yet at the switch side. Therefore, an understanding on the potential of auxiliary connections in improving network performance is a prerequisite for its real-world data center deployment. In this context, an analytical model which produces evaluation results with realistic data center traffic loads is indispensable.

In this paper, we develop a queueing model for auxiliary-connection-enabled switches with limited buffer space. The model captures not only the *packet_in* interaction, but also the *stats/feature_request/reply* interactions, which are of great importance to enable SDN benefits, such as efficient resource allocation. Equations are derived to evaluate the network performance. The model is validated for lower traffic loads using the network simulator ns-3 [6], where network protocols including OpenFlow are implemented. Our model is then used to perform an in-depth analysis of switch performance under realistic data center traffic loads. In particular, we thoroughly characterize the packet loss rate, average number of packets in switch, and flow setup delay in the network. With limited hardware resource available at switch to build auxiliary connections, our model can serve as a tool to predict network performance and provision switches with the appropriate number of connections to satisfy the network QoS requirements. To the best of our knowledge, this is the first analytical model of auxiliary-connection-enabled OpenFlow switches. The rest of the paper is organized as follows. The background is provided in Section II. The queueing model is developed and validated in Section III. Moreover, the network performance evaluation is provided in Section IV. Finally, the paper is concluded in Section V.

II. Background

A. OpenFlow-based SDN and Auxiliary Connections

As the *de facto* southbound protocol of SDN, OpenFlow enables the communication between the controller and switch. When a switch receives a packet from a host, it iterates through its flow tables to look for matched flow entry. If the switch cannot find a matched entry, it sends a *packet_in* message to the controller via the switch-to-controller link. It can either send only the packet header with the payload buffered at the switch, or send the whole packet to the controller without buffering it depending on the switch implementation. The first approach suffers from the limited switch buffer size, while the second one suffers

978-1-5386-9223-3/19/$31.00 ©2019 IEEE

from the limited bandwidth of the switch-to-controller link. With auxiliary connections implemented and the switch-to-controller link limitation addressed, the second approach is advantageous since it avoids switch buffer overflow. Therefore in this work, we focus on the second approach, which is utilized by Open vSwitch 2.7 and after. Once the controller receives the *packet_in* message, it will instruct the switch to forward the incoming packets and install a new flow entry. Besides this *packet_in* interaction, the controller also sends messages like *feature/stats_request* to OpenFlow switches periodically to query necessary information for network management. The replies from OpenFlow switches are *feature/stats_reply*.

By default, the channel between an OpenFlow switch and controller is a single network connection. In order to exploit the parallelism of switch implementation, OpenFlow 1.3 proposed to create auxiliary connections besides the main connection between switches and the controller. According to the specification, the controller is free to use the various switch connections for sending OpenFlow messages at its entire discretion. Thus we assume that the auxiliary connections behave the same as the main connection and they are all referred to as multiple connections in this work.

B. Related Work

Analytical models have been developed for SDN [7]–[12], but they are limited in the following aspects. First, none of them considers auxiliary connections in their model. Second, the only interaction considered in these works is the *packet_in* process. They omit the *feature/stats_request/reply* interactions. Furthermore, only [10] considers a realistic OpenFlow switch with limited buffer space. Without a realistic assumption of the OpenFlow switch, the models will not be able to capture the accurate network performance, such as packet loss rate. Finally, the validation results are not provided in [7]. In this context, we propose our model to compensate for these limitations.

III. Analytical Model

In this section we develop and validate the queueing model of auxiliary-connection-enabled switches as shown in Fig.1.

A. Queueing Model

The switch is modeled as a two-node queueing network composing of $S0$ and $S1$. Node $S0$ collects all the incoming messages to the switch while $S1$ sends messages to the controller. External packets from the network arrive at $S0$ according to a Poisson process with rate λ_S. Its service time is exponentially distributed with rate μ_{S0}. We assume that the probability of packets not being sent to controller is β, e.g., packets from the matched flows. The messages need to be sent to the controller, e.g., packets from the unmatched flows, will be sent to $S1$. The multiple connections between the switch and controller are modeled as W servers of $S1$, with exponentially distributed service times at rates $\mu_1, \mu_2 \cdots \mu_W$. The total buffer space at the switch is limited at N and packet loss will happen if the buffer space is full.

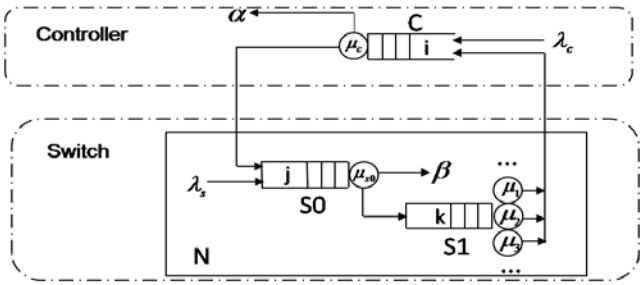


Fig. 1. Queueing model of auxiliary-connection-enabled switches for SDN. The switch is modeled as a two-node queueing network composing of $S0$ and $S1$. The controller is modeled as a single queue node C.

The controller is modeled as a single queue node C with service time, following exponential distribution at rate μ_C, which includes both the controller processing time and the transmission time to switch, and is positively correlated with W since the transmission time is dependent on the number of connections. The incoming traffic to C is made up of the messages from the switch, (such as *packet_in*, *feature/stats_reply*) and the controller-generated messages (such as *flow_mod, feature/stats_request*). The second type of messages is generated according to a Poisson process with rate λ_C. The incoming traffic to C either causes a corresponding message to be sent to switch, (e.g., a *packet_out* message will be sent to the switch for each *packet_in* message) or will be finished at the controller (e.g., the controller collects the statistics with the *stats_reply* messages). We assume the probability that the incoming messages are finished at controller is α. Parameter λ_C and α varies based on the number of *feature/stats_request/reply* messages in the network, and is used to model these interactions. Since SDN controller is implemented on a high-power server, we assume the buffer space of C is unlimited.

B. Model Analysis

This queueing model can be analyzed as a three-dimensional continuous time Markov chain (CTMC) using quasi-birth-death process [13]. Its multidimensional state space $\mathbf{T}$ is given as set of tuples:

$$\mathbf{T} = \{(i, j, k) | i, j, k \in \mathbb{N}_0, j + k \leq N\},$$

where i, j, k is the number of packets in node $C, S0$, and $S1$. With i as the level variable, the transition rate matrix of the CTMC can be grouped into finite sub-matrices with block structures: B_0, A_0, A_1, and A_2, which is given as:

$$\mathbf{Q} = \begin{array}{c} \\ T_0 \\ T_1 \\ T_2 \\ T_3 \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccc} T_0 & T_1 & T_2 & T_3 & \cdots \end{array} \\ \begin{pmatrix} B_0 & A_0 & \mathbf{0} & \mathbf{0} & \cdots \\ A_2 & A_1 & A_0 & \mathbf{0} & \cdots \\ \mathbf{0} & A_2 & A_1 & A_0 & \cdots \\ \mathbf{0} & \mathbf{0} & A_2 & A_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \end{array},$$

where T_i is a subset of $\mathbf{T}$ with the same i value. Moreover, B_0 is the transition rate matrix within level $i = 0$, and A_2, A_1, A_0 are transition rate matrices from level i to level

$i-1$, within level i, and from level i to level $i+1$, respectively. Taking j as the second level variable, each of the matrices can be further grouped into sub-matrices to determine its elements.

1) A_0: If the value of i increases by 1, the incoming packet at C can come from the controller-generated messages or from the switch, and thus the value of k does not change or decreases by 1, respectively. The value of j does not change in both cases. Therefore, A_0 can be structured as:

$$A_0 = \begin{array}{c} \\ T_{i,0} \\ T_{i,1} \\ T_{i,2} \\ \vdots \\ T_{i,N} \end{array} \begin{array}{c} \begin{array}{ccccc} T_{i+1,0} & T_{i+1,1} & T_{i+1,2} & \cdots & T_{i+1,N} \end{array} \\ \begin{pmatrix} C^0 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & C^1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & C^2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & C^N \end{pmatrix} \end{array}.$$

Matrix C^j is dependent on the second level variable j.

$$C^j_{(k,k')\in\mathbb{N}_{N-j}\times\mathbb{N}_{N-j}} = \begin{cases} \lambda_C, & \text{if } k' = k \\ \mu_{S1}, & \text{if } k' = k-1 \\ 0, & \text{Otherwise} \end{cases},$$

where $\mathbb{N}_n$ denotes the set of natural numbers $\{0, 1, \cdots n\}$, and μ_{S1} is the overall service rate at queue node $S1$:

$$\mu_{S1} = \begin{cases} \mu_1 + \cdots + \mu_k, & \text{if } k < W \\ \mu_1 + \cdots + \mu_R, & \text{Otherwise} \end{cases}.$$

2) A_2: If the value of i decreases by 1, the outgoing message either goes to $S0$ or is finished at C, and thus the value of j increases by 1 or does not change, respectively. The value of k does not change in both cases. Thus A_2 is:

$$A_2 = \begin{array}{c} \\ T_{i,0} \\ T_{i,1} \\ T_{i,2} \\ \vdots \\ T_{i,N} \end{array} \begin{array}{c} \begin{array}{ccccc} T_{i-1,0} & T_{i-1,1} & T_{i-1,2} & \cdots & T_{i-1,N} \end{array} \\ \begin{pmatrix} D^0 & E^0 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & D^1 & E^1 & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & D^2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & D^N \end{pmatrix} \end{array},$$

where

$$D^j_{(k,k')\in\mathbb{N}_{N-j}\times\mathbb{N}_{N-j}} = \begin{cases} \alpha\mu_C, & \text{if } k' = k \\ 0, & \text{Otherwise} \end{cases},$$

$$E^j_{(k,k')\in\mathbb{N}_{N-j}\times\mathbb{N}_{N-j-1}} = \begin{cases} (1-\alpha)\mu_C, & \text{if } k' = k \\ 0, & \text{Otherwise} \end{cases}.$$

3) A_1: If the value of i does not change, the value of j can increase by 1, decrease by 1, or do not change. Thus, A_1 is structured as following:

$$A_1 = \begin{array}{c} \\ T_{i,0} \\ T_{i,1} \\ T_{i,2} \\ \vdots \\ T_{i,N} \end{array} \begin{array}{c} \begin{array}{ccccc} T_{i,0} & T_{i,1} & T_{i,2} & \cdots & T_{i,N} \end{array} \\ \begin{pmatrix} G^0 & H^0 & \mathbf{0} & \cdots & \mathbf{0} \\ F^1 & G^1 & H^1 & \cdots & \mathbf{0} \\ \mathbf{0} & F^2 & G^2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & G^N \end{pmatrix} \end{array}.$$

If i does not change and j decreases by 1, the output message of $S0$ can either be forwarded to $S1$ with probability $1-\beta$, or be finished at switch with probability β. Thus

$$F^j_{(k,k')\in\mathbb{N}_{N-j}\times\mathbb{N}_{N-j+1}} = \begin{cases} \beta\mu_{S0}, & \text{if } k' = k \\ (1-\beta)\mu_{S0}, & \text{if } k' = k+1 \\ 0, & \text{Otherwise} \end{cases}.$$

Next, we derive matrix H^j, where j increases by 1 and k does not change. The incoming packet to $S0$ is from the external network since i does not change.

$$H^j_{(k,k')\in\mathbb{N}_{N-j}\times\mathbb{N}_{N-j-1}} = \begin{cases} \lambda_S, & \text{if } k' = k \\ 0, & \text{Otherwise} \end{cases}.$$

The matrix $G^j_{(k,k')\in\mathbb{N}_{N-j}\times\mathbb{N}_{N-j}}$ is a diagonal matrix, whose elements stand for the transition rates where the values of i, j, k do not change. It ensures that all elements in each row of $\mathbf{Q}$ sum up to 0. Its diagonal elements are:

$$G^j_{(k,k)} = \begin{cases} -\lambda_C - \lambda_S - \mu_C, & \text{if } j = 0 \text{ and } k = 0 \\ -\lambda_C - \lambda_S - \mu_C - \mu_{S1}, & \text{if } j = 0 \text{ and } 0 < k < N \\ -\lambda_C - \lambda_S - \mu_C - \mu_{S0}, & \text{if } k = 0 \text{ and } 0 < j < N \\ -\lambda_C - \mu_{S1} - \alpha\mu_C, & \text{if } j = 0 \text{ and } k = N \\ -\lambda_C - \mu_{S0} - \alpha\mu_C, & \text{if } k = 0 \text{ and } j = N \\ -\lambda_C - \mu_{S1} - \alpha\mu_C - \mu_{S0}, & \text{if } 0 < j < N \text{ and } k = N - j \\ -\lambda_C - \lambda_S - \mu_C - \mu_{S1} - \mu_{S0}, & \text{if } 0 < j < N \text{ and } 0 < k < N - j \end{cases}$$

4) B_0: Matrix B_0 has the same structure, and sub-matrices $F^{0,j}, H^{0,j}$ as F^j, H^j in A_1. The only difference is the diagonal matrix $G^{0,j}$, where

$$G^{0,j}_{(k,k)} = \begin{cases} -\lambda_C - \lambda_S, & \text{if } j = 0 \text{ and } k = 0 \\ -\lambda_C - \lambda_S - \mu_{S1}, & \text{if } j = 0 \text{ and } 0 < k < N \\ -\lambda_C - \lambda_S - \mu_{S0}, & \text{if } k = 0 \text{ and } 0 < j < N \\ -\lambda_C - \mu_{S1}, & \text{if } j = 0 \text{ and } k = N \\ -\lambda_C - \mu_{S0}, & \text{if } k = 0 \text{ and } j = N \\ -\lambda_C - \mu_{S1} - \mu_{S0}, & \text{if } 0 < j < N \text{ and } k = N - j \\ -\lambda_C - \lambda_S - \mu_{S1} - \mu_{S0}, & \text{if } 0 < j < N \text{ and } 0 < k < N - j \end{cases}$$

This CTMC process is stable if $\pi_A A_0 \mathbf{1} < \pi_A A_2 \mathbf{1}$, where π_A is the steady-state probability vector of the generator matrix $A = A_0 + A_1 + A_2$, and $\mathbf{1}$ is a column vector of 1's. With a stable CTMC and the transition rate matrix $\mathbf{Q}$ constructed, the steady-state probability vector π can be calculated. First, we partition π according to the level variable i: $\pi = [\vec{\pi}_0, \vec{\pi}_1, \cdots]^T$, where $\vec{\pi}_i = [\pi_{i,0,0}, \pi_{i,0,1}, \cdots, \pi_{i,0,N}, \pi_{i,1,0}, \cdots, \pi_{i,1,N-1}, \cdots, \pi_{i,N,0}]$. The global balance equations for $i > 0$ is:

$$\vec{\pi}_{i-1}A_0 + \vec{\pi}_i A_1 + \vec{\pi}_{i+1}A_2 = 0. \quad (1)$$

Since $\vec{\pi}_i$ can be defined in terms of $\vec{\pi}_{i-1}$ and the transitions between the levels are independent of level i, a constant rate matrix R is defined to lead to the matrix-geometric equation:

$$\vec{\pi}_i = \vec{\pi}_{i-1}R = \vec{\pi}_1 R^{i-1}, i > 0. \quad (2)$$

Substitute Equation (2) into Equation (1), we can get

$$A_0 + RA_1 + R^2A_2 = 0 \quad (3)$$

to solve R. Finally, the steady-state probability vector $\vec{\pi}_i$ can be obtained using R, the boundary condition:

$$\vec{\pi}_0 B_0 + \vec{\pi}_1 A_2 = 0,$$
$$\vec{\pi}_0 A_0 + \vec{\pi}_1 A_1 + \vec{\pi}_2 A_2 = 0,$$

and the fact that the sum of all elements in π is 1.

C. Performance Analysis

With the steady-state probability calculated, we derive the equations for main network performance metrics, including the packet loss rate (P_S), average number of packets in switch (L_S), and the flow setup delay (T_{setup}).

The average departure rates at $C, S0$, and $S1$ are:

$$TH_C = \mu_C(1 - \sum_{(0,j,k)\in T} \pi_{0,j,k}),$$
$$TH_{S0} = \mu_{S0}(1 - \sum_{(i,0,k)\in T} \pi_{i,0,k}),$$
$$TH_{S1} = \mu_{S1}(1 - \sum_{(i,j,0)\in T} \pi_{i,j,0}).$$

In stationary state, the incoming and outgoing traffic at switch obeys the following relation:

$$[\lambda_S + (1-\alpha)TH_C](1 - P_S) = \beta * TH_{S0} + TH_{S1},$$

and thus the loss rate at switch can be derived. The average number of customers in each queue node (L_C, L_{S0}, L_{S1}) and in the switch (L_S) is:

$$L_C = \sum_{(i,j,k)\in T} i \times \pi_{i,j,k},$$
$$L_{S0} = \sum_{(i,j,k)\in T} j \times \pi_{i,j,k},$$
$$L_{S1} = \sum_{(i,j,k)\in T} k \times \pi_{i,j,k},$$
$$L_S = L_{S0} + L_{S1}.$$

The flow setup delay is defined as the time interval between the time that the first packet of a new flow arrives at the switch and the time it is forwarded by the switch to network. It is composed of the queueing delays at the switch and controller and the propagation delay. Based on Little's Law,

$$T_{\text{setup}} = \frac{L_{S0}}{TH_{S0}} + \frac{L_{S1}}{TH_{S1}} + \frac{L_C}{TH_C} + \frac{L_{S0}}{TH_{S0}} + T_{prop},$$

where T_{prop} is the two-way propagation delay.

TABLE I
PARAMETER SETTINGS

Parameters	Validation	Evaluation
λ_C, μ_C [packets/s]	$70, 220W$	$10k, 120000W$
λ_S [packets/s]	$25 - 370$	$100k - 1000k$
μ_{S0} [packets/s]	2000	$1000k$
$\mu_1, \mu_2, \cdots$ [packets/s]	$100, 100, \cdots$	$50k, 50k, \cdots$
W, N	$[1, 2, 3]$, 20	$[1, 2, 3], 10 - 120$
α, β	$1\%, 80\%$	$10\%, 87\% - 99\%$

D. Multiple-Switch Network

To utilize the switch model in network with M switches, we assume that for a certain message sent from controller to switch, the probability that the destination is switch $1, 2, \cdots, M$ is $p_1, p_2, \cdots, p_M$ and $\sum_{m=1}^{M} p_m = 1$. Thus, the transition rate where switch m receives a packet from controller is $p_m(1-\alpha)\mu_C$, and transition rate where C receives a packet from switches is $\sum_{\text{m=1}}^{M} \mu_{S1}^m$. The network becomes a $2M+1$ dimensional CTMC and the performance of each switch can be analyzed following similar procedure as presented above. Further exploration on the multiple-switch case will be conducted in the future.

E. Validation through Simulation

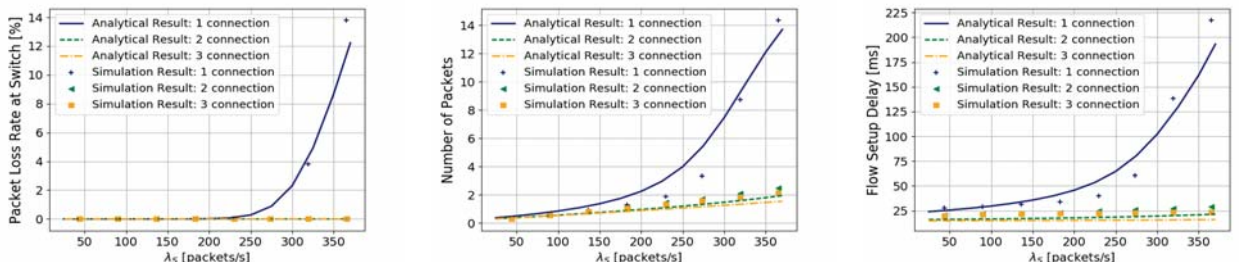
We first confirm the mathematical solution using Python simulation [14] and the parameters are in the "Validation" column of Table I. The simulated queueing model reflects the structure shown in Fig.1. Our analytical results for P_S, L_S and T_{setup} matched exactly with the simulation results validating the correctness of our analyses.

In order to validate our queueing model in a realistic network environment, we used ns-3 to simulate an SDN network with OpenFlow and auxiliary connections implemented [15]. There is one OpenFlow switch with varying number of connections to the controller. There are 200 hosts attached to the switch and each host starts to send traffic randomly. In the simulation, we use a realistic traffic generator on each host to relax the assumption of the Poisson arrival process used in the analytical model. The traffic is ON/OFF and the bitrate is constant. The actual packet inter-arrival rate to switch (λ_S) is calculated based on the simulation results. The "Validation" column of Table I lists some additional parameters. The results are shown in Fig.2, where each data point is obtained based on ten simulation runs using different random seeds. The simulation results match well with the theoretical ones indicating that our analytical model is able to capture the trend of performance changes.

In next section, we evaluate performance for a switch with a varying number of connections using the parameters in the "Evaluation" column of Table I. Note that the packet inter-arrival rate is more than 3 orders of magnitude larger than the value used in this section. With this larger inter-arrival rate, Python simulation would take several days, the ns-3 simulation would need almost one week, but evaluation using our analytical model required only seconds.

IV. NETWORK PERFORMANCE EVALUATION

In this section, we use our analytical model to evaluate switch performance with 1–3 switch-to-controller connec-



(a) (b) (c)

Fig. 2. Validation for analytical model: (a) packet loss rate, (b) average number of packets in switch, (c) average flow setup delay

tions. The parameter settings are from the "Evaluation" column in Table I, and we vary λ_S, $1-\beta$ (probability that an incoming packet goes to the controller), and N. Evaluations such as these can be used to provision switches with an appropriate number of connections based on the expected traffic characteristics within a deployed network.

A. Varying Packet Arrival Rate

As shown in Fig.3(a), the packet loss rate of the single-connection case increases fast when $\lambda_S > 400k$ and reaches around 50% when $\lambda_S = 900k$. On the other hand, the loss rate does not increase until λ_S reaches $850k$ packets/s for the multiple-connection cases. The three-connection case outperforms the two-connection case but the improvement (around 1%) is minimal. In this case, the capacity limitation exists in $S0$, so increasing the number of connections in $S1$ cannot improve the performance significantly.

We use the buffer utilization level to indicate the average number of packets stored in switch (shown in Fig.3(b)). With a smaller packet arrival rate ($\lambda_S < 400k$), the switch buffer utilization level stays at a relatively low level for all cases, which corresponds with a zero packet loss rate. The switch buffer becomes more utilized as λ_S increases, and finally is bounded by the buffer size limitation. A high utilization level indicates that the switch is full most of the time and thus high loss rate is caused. When the utilization level is greater than 10%, all the cases start to induce packet loss.

Fig.3(c) shows the average flow setup delay. Multiple connections can improve the flow setup delay by 37-50% with varying λ_S. According to the queueing theory, the queueing delay depends on the buffer size. Thus when the buffer utilization level is greater than 90%, the flow setup delay of the single-connection case stays at a fixed value.

B. Varying $1-\beta$

The value of $1-\beta$ depends on the network traffic characteristics. For example, if the average flow length in the network is low, the probability that a packet is the first packet of a flow and needs to go to the controller is high, which means $1-\beta$ is high. In general, when the probability of the incoming packets going to the controller increases, the performance (as shown in Fig.4) persists a similar trend as λ_S increases since they both burden $S1$. However, while the value of λ_S impacts the overall load on the switch, including both $S0$ and $S1$, the change of $1-\beta$ will induce different load to $S1$ specifically. Thus, a higher processing rate of $S1$ can improve the performance significantly. The performance improvement induced by utilizing more connection is much more obvious compared with the case when we vary λ_S. For example, when $1-\beta = 12\%$, the three-connection case induces a notably better performance for all the metrics than the two-connection case.

C. Varying Switch Buffer Space

As shown in Fig.5(a), as the switch buffer size increases, the multiple-connection cases can decrease the loss rate by 30-45% compared with the single-connection case and achieve a stable service. With varying buffer space available in the switch, the loss rate with single-connection case stays at 50%, indicating more buffer space is required to provide a stable service. In general, with more connections in use, the switch requires less buffer to provide a stable service.

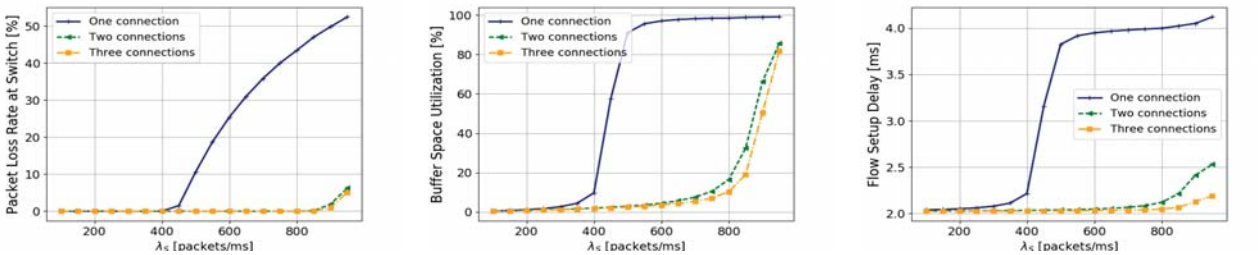
In terms of buffer utilization level as shown in Fig.5(b), the switch with only one connection tends to be fully occupied all the time, and thus the packet loss rate stays high. Cases with more connections always generate a lower utilization level. For example, the three-connection case incurs around 10% less utilization than the two-connection case.

Fig.5(c) shows that the flow setup delay of the single-connection case increases linearly with the switch buffer size. This linear increase can be justified by queueing theory, which states that a larger buffer size allows more packets waiting in the queue and thus may cause a longer delay. With only one connection, more buffer space does not help with the packet loss rate, but generates a longer flow setup delay instead. On the other hand, with multiple connections, the switch maintains a low buffer utilization level, and thus the flow setup delay stays at a lower level.

In summary, with different traffic characteristics, different number of connections are recommended to improve the network performance. Our model can serve as a tool to find the minimum number of required connections.

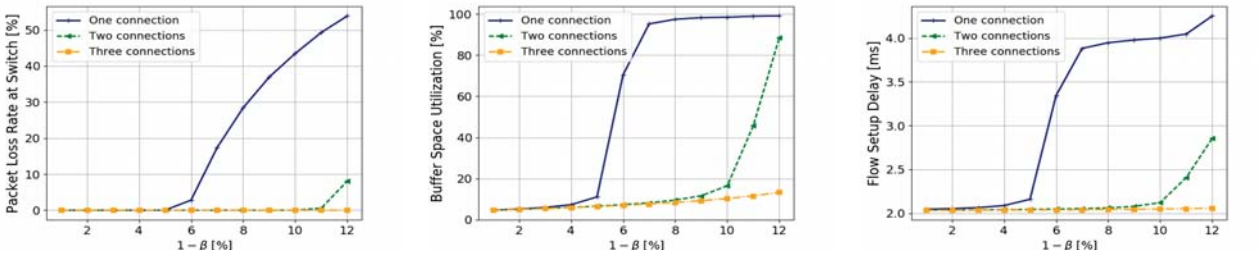
V. Conclusion

In this paper, we propose and validate an analytical queueing model for auxiliary-connection-enabled OpenFlow switches. This is the first known work to analytically study auxiliary connections in SDN. Our analysis can be used in the network deployment phase to provision switches with the appropriate number of auxiliary connections based on expected traffic characteristics.



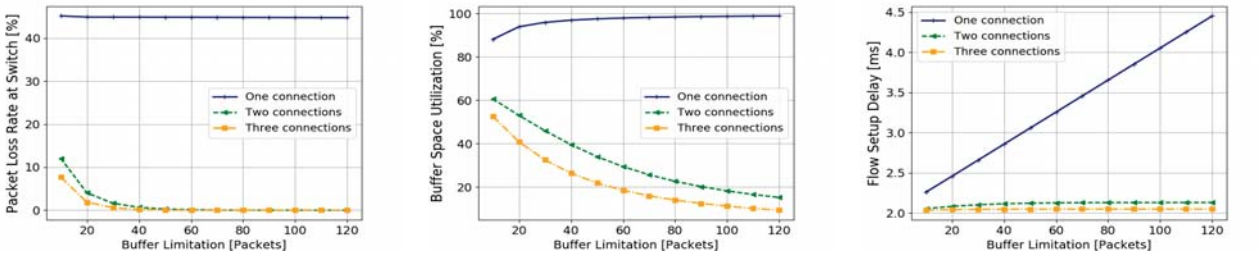
(a) (b) (c)

Fig. 3. Network performance with varying λ_S, $N = 100$, $\beta = 90\%$ (a) packet loss rate, (b) switch buffer utilization, (c) flow setup delay



(a) (b) (c)

Fig. 4. Network performance with varying $1 - \beta$, $\lambda_S = 800k$, $N = 100$ (a) packet loss rate, (b) switch buffer utilization, (c) flow setup delay



(a) (b) (c)

Fig. 5. Network performance with varying N, $\lambda_S = 800k$, $\beta = 90\%$ (a) packet loss rate, (b) switch buffer utilization, (c) flow setup delay

REFERENCES

[1] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 267–280.

[2] A. Bianco, P. Giaccone, A. Mahmood, M. Ullio, and V. Vercellone, "Evaluating the sdn control traffic in large isp networks," in *Communications (ICC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5248–5253.

[3] H. Huang, S. Guo, W. Liang, K. Li, B. Ye, and W. Zhuang, "Near-optimal routing protection for in-band software-defined heterogeneous networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 11, pp. 2918–2934, 2016.

[4] The Open Networking Foundation, "OpenFlow Switch Specification," Jun. 2012.

[5] L. F. Müller, R. R. Oliveira, M. C. Luizelli, L. P. Gaspary, and M. P. Barcellos, "Survivor: an enhanced controller placement strategy for improving sdn survivability," in *Global Communications Conference (GLOBECOM), 2014 IEEE*. IEEE, 2014, pp. 1909–1915.

[6] "ns-3 a discrete-event network simulator ns-3.16," https://www.nsnam.org/docs/release/3.16/doxygen/index.html.

[7] 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 *Global Communications Conference (GLOBECOM), 2013 IEEE*. IEEE, 2013, pp. 1397–1402.

[8] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an openflow architecture," in *Proceedings of the 23rd international teletraffic congress*. International Teletraffic Congress, 2011, pp. 1–7.

[9] K. Sood, S. Yu, and Y. Xiang, "Performance analysis of software-defined network switch using $m/geo/1$ model," *IEEE Communications Letters*, vol. 20, no. 12, pp. 2522–2525, 2016.

[10] Y. Goto, H. Masuyama, B. Ng, W. K. Seah, and Y. Takahashi, "Queueing analysis of software defined network with realistic openflow–based switch model," in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2016 IEEE 24th International Symposium on*. IEEE, 2016, pp. 301–306.

[11] 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.

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

[13] G. Bolch, S. Greiner, H. De Meer, and K. S. Trivedi, *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[14] P. Geraint, K. Vince, Lieke19, L. Sam, caipirginka, T. G. Badger, Nikoleta, C. Alex, and J. Adam, "Ciw: v1.1.5," 2018.

[15] H. Yang, C. Zhang, and G. Riley, "Support multiple auxiliary tcp/udp connections in sdn simulations based on ns-3," in *Proceedings of the Workshop on ns-3*. ACM, 2017, pp. 24–30.