An Analytical Model for Software Defined Networking: A Network Calculus-based Approach

Siamak Azodolmolky*, Reza Nejabati‡, Maryam Pazouki*†, Philipp Wieder*,
Ramin Yahyapour* and Dimitra Simeonidou‡
*Gesellschaft für Wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG), Göttingen 37075, Germany,
E-mail: {Siamak.Azodolmolky, Maryam.Pazouki, Philipp.Wieder, Ramin.Yahyapour}@gwdg.de
†Institut für Numerische und Angewandte Mathematik,Universität Göttingen, Göttingen 37083, Germany,
‡Department of Electrical & Electronic Engineering, University of Bristol, Clifton BS8 1UB, UK,
E-mail: {Reza.Nejabati, Dimitra.Simeonidou}@bristol.ac.uk

***Abstract*—Software defined networking (SDN) and OpenFlow as the outcome of recent research and development efforts provided unprecedented access into the forwarding plane of networking elements. This is achieved by decoupling the network control out of the forwarding devices. This separation paves the way for a more flexible and innovative networking. While SDN concept and OpenFlow find their ways into commercial deployments, performance evaluation of the SDN concept and its scalability, delay bounds, buffer sizing and similar performance metrics are not investigated in recent researches. In spite of usage of benchmark tools (like OFlops and Cbench), simulation studies and very few analytical models, there is a lack of analytical models to express the boundary condition of SDN deployment. In this work we present a model based on network calculus theory to describe the functionality of an SDN switch and controller. To the best of our knowledge, this is for the first time that network calculus framework is utilized to model the behavior of an SDN switch in terms of delay and queue length boundaries and the analysis of the buffer length of SDN controller and SDN switch. The presented model can be used for network designers and architects to get a quick view of the overall SDN network deployment performance and buffer sizing of SDN switches and controllers.**

Keywords*—*Software Defined Networking (SDN), OpenFlow, Network Calculus, Delay Bound, Buffer Sizing

I. Introduction

Software defined networking (SDN) (to mention a few [1], [2], [3], [4], and [5]) is a concept, which allows network operators to flexibly manage routers and switches using software running on external servers. While the term SDN can apply to many network designs with decoupled control and data planes, OpenFlow [5] is one of the early implementations of SDN. In OpenFlow each switch within the network exports an interface that provides a remote controller to manage its forwarding tables that allow mapping between packet header fields and instructions to execute on matching packets. OpenFlow, and SDN functionalities in general have been studied in the recent years, however the question remains open how to evaluate the performance of SDN and OpenFlow deployments.

Currently, first OpenFlow implementations from hardware vendors are being deployed in networks and a growing number of experiments over SDN-enabled networks are expected. This will create new challenges, as question of SDN performance and scalability have not yet been properly investigated. Understanding the performance and limitation of the SDN concept is a requirement for its implementation in practical applications. There are very few performance evaluation studies of OpenFlow and SDN architecture. Besides, in order to have an initial estimate of the performance and requirement of an SDN deployment, is understandably essential for network architects and designers. Although simulation studies and experimentation are among the widely used performance evaluation techniques, analytical modeling has its own benefits. A closed-form description of a networking architecture paves the way for network designers to have a quick (and approximate) estimate of the performance of their design, without the need to spend considerable time for simulation studies or expensive experimental setup.

In line with our recent research activities to experimentally demonstrate different SDN schemes [6], [7], in this work we utilize network calculus as a mathematical framework to analytically model the behavior of an SDN switch and its interaction with an SDN controller. To the best of our knowledge, this is for the first time that a network calculus-based analytical model is investigated and presented to model an SDN architecture.

After this introduction, related studies to evaluate the performance of OpenFlow and SDN framework are compiled in Section II. In particular recent studies, and shortcoming of classic queuing theory for analytical modeling is discussed in this section. Network calculus framework and detailed description of our models is presented in III. This section includes an overview on network calculus, definitions, system model, and analysis of an SDN switch and the interaction of switches with SDN controllers. The mathematical description of queue length and delay bound of SDN switches along with the buffer requirements of the SDN controller and SDN switches in a deployment is presented in this section. Using the results of some recent evaluations, we present the boundary performance of packet delay in switches and buffer sizing of an SDN controller in Section IV. Finally we draw our conclusions and future research in Section V.

II. Related work

In spite of active research around SDN and OpenFlow in particular, there are very few works to either address the

978-1-4799-1353-4/13/$31.00 ©2013 IEEE

performance issue of the SDN and OpenFlow as one of its early implementation or evaluation frameworks. As the building blocks of the SDN and OpenFlow ecosystem, the performance and analytic behavior of SDN controllers and switches are very important for the entire network. Thus, it is important to identify and properly evaluate the performance metrics of SDN (or OpenFlow)-based deployment.

An architectural design to improve the lookup performance of OpenFlow switching in Linux operating system and using a standard commodity network interface card is investigated in [8]. Authors in [8] reported a packet switching throughput improvement up to 25% compared to the one based on a soft OpenFlow switching. Hardware acceleration based on network processors are applied in [9] to perform OpenFlow switching. They report a 20% performance improvement in terms of packet delay, compared to the conventional designs [9]. The performance of the OpenFlow switch is measured for different types of rules and packet sizes in [10]. However, the interaction of switch and controller is not considered in the mentioned work. Inherent performance bottlenecks with respect to the CPU load in current OpenFlow switch implementations were observed and reported in [11] and modifications to the OpenFlow protocol is proposed. Authors in [12] investigated the usability of OpenFlow in data centers. They discovered that reactive setting of flow rules leads to an unacceptable performance when only eight switches are handled by one OpenFlow controller. An open framework (i.e., OFlops [13]) for performance analysis of OpenFlow switches exists, however on the controller side only a relatively simple benchmark tool (i.e., Cbench [14]) exists. Cbench has become the standard evaluation tool for controller performance, which we also used in our previous work [6] to evaluate the performance of an extended OpenFlow controller.

Possible performance improvement of OpenFlow controllers are highlighted in [15] using Cbench tool. Cbench is a single-threaded tool and therefore multiple instances have to be started to utilize multiple CPUs. It also only establishes one controller connection for all emulated switches. Therefore, little can be derived from the results in terms of controller performance and behavior or estimation of different bounds. For instance, aggregated statistics are gathered for all switches but not for each individual switch. As a result, it is not possible to identify whether all responses of the controller are for a single switch and the others receive null, or whether the capacity of the controller is shared among the switches. A flexible OpenFlow controller benchmark tool is introduced in [16]. It creates a set of message-generating virtual switches, which can be configured independently from each other to emulate a specific scenario and to maintain their own statistics.

SDN and OpenFlow initiate new challenges, as questions of scalability and performance of SDN (and OpenFlow), have not yet properly addressed. A basic analytic model based on queuing theory for the forwarding speed and blocking probability of current OpenFlow switches are proposed in [8]. However, the model does not capture the fact that incoming traffic at the switch is first queued per line card. It is also limited to a single switch per controller, where as an SDN architecture allows the same controller to be contacted by a number of different switches. The advantage of an analytical model is that it can quickly provide performance indicators and potential scalability bottlenecks for an OpenFlow switch-controller system before detailed data is available. While simulation can provide detailed insight into a certain configuration, the analytical model greatly simplifies a conceptual deployment decision.

There are two important approaches towards analytical performance evaluation of computer networks: queuing theory [17] and network calculus [18]. The classical queuing theory generally concerns about the average quantities in equilibrium at the steady state, and accuracy is its target in the analysis. Therefore, obtaining a right set of tractable results comes at the cost of having to restrict to Markovian (memoryless) traffic, which is not necessarily a realistic assumption for Ethernet traffic [19]. In contrast to queuing theory, network calculus is concerned with worst case (upper bounds) instead of average (equilibrium) behavior and therefore does not deal with arrival and departure processes themselves but with bounding processes called arrival and service curves. A comprehensive overview and outlook of stochastic network calculus can be found in [20]. By focusing on bounds, network calculus compliments the classical queuing theory. To the best of our knowledge, this is for the first time that a network calculus-based analytical study is presented to model the behavior of SDN. Our analytical model provides a mathematical framework of the system, which describe the SDN ecosystem and its high-level performance, which is detailed in next sections.

III. Analytical Modeling Framework

There are two level of services in an SDN-based network architecture: flow-level and packet-level. While for each SDN switch (e.g., OpenFlow switch), packets are the basic serviced units, for the SDN-controller, the flows corresponding to the users' applications are the serviced units. Considering both packet-level and flow-level arrival processes in the network implies unique arrival and service characteristics and requirements. Specifically, for the arrival process a flow-level arrival, usually consists of multiple and possibly many packet-level arrival processes.

In the classical queuing theory, normally only one level of packets is considered in its arrival and service process modeling. Besides, packet inter-arrival time and service times generally do not follow the exponential distribution assumption in an Ethernet network [19]. Furthermore, from a network designer point of view, it is better to have an overview of boundary conditions rather than average (equilibrium) state of the network. All these have made performance evaluation of SDN architecture challenging.

A. Network Calculus Background

Network calculus is a tool to analyse flow control problems in networks mainly from a lower bound (i.e., worst case) perspective. In other words it is a framework to derive deterministic guarantees on delay, queue lengths, throughput, and similar performance metrics. It is mathematically based on min-plus algebra, which could also be interpreted as a system theory for deterministic and stochastic queuing systems. Most of the network flows can be described using arrival curves, represented with leaky bucket traffic envelopes and most of the

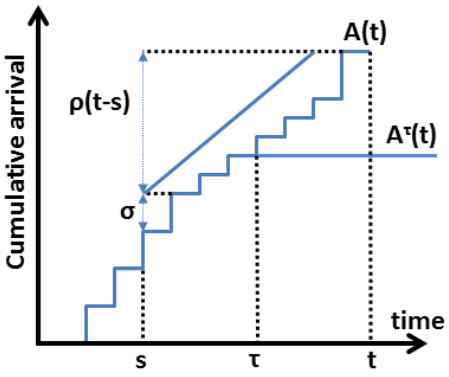

Fig. 1. Graphical representation of cumulative arrival process $A(t)$ with average sustainable arrival rate of ρ and burstiness of σ; along with a stopped sequence $A^{\tau}(t)$.

network elements provide some service to the flows, described by its rate and slack term [21], [22].

The use of alternate algebra such as min-plus algebra and max-plus algebra to transform complex network systems into analytically tractable models is central to the network calculus theory. Furthermore, in order to simplify the analysis, arrival and service processes are characterized by some bounds and the performance evaluations are based on such bounds.

B. Definitions

Network calculus uses traffic specification to model the arrival and peak characteristics of a flow. A cumulative arrival process A is a non-decreasing, integer valued function on the non-negative integer Z_+ such that $A(0) = 0$. $A(t)$ denotes the total number of packet arrivals in time slots $1, 2, ..., t$. The burstiness and the average sustainable rate of arrivals are represented by σ and ρ respectively. The number of packet arrivals at time t is denoted by $a(t)$ and $a(t) = A(t) - A(t-1)$. The cumulative arrival process (A) is said to be (σ, ρ)-upper constrained, which we denote it as $A \sim (\sigma, \rho)$, if (see Fig. 1):

$$A(t) - A(s) \leq \sigma + \rho(t-s), 0 \leq s \leq t. \quad (1)$$

According to the multiplexing rule [18], if constrained flows are merged, the output process is also constrained or:

$$A_i \sim (\sigma_i, \rho_i) \rightarrow \sum A_i \sim (\sum \sigma_i, \sum \rho_i) \quad (2)$$

For any increasing sequence A (i.e., cumulative arrival process), we define its "stopped sequence" at time τ (see Fig. 1), denoted as A^{τ}, by:

$$A^{\tau}(t) = \begin{cases} A(t) & \text{if } t \leq \tau, \\ A(\tau) & \text{otherwise} \end{cases}. \quad (3)$$

According to (3) for the stopped sequence A^{τ}, if A is an arrival process, then there are no further packet arrival after time τ. We can simply show that a stopped sequence A^{τ} is $(\sigma(\tau), \rho)$-upper constrained where

$$\sigma(\tau) = \max_{0 \leq t \leq \tau} \max_{0 \leq s \leq t} [A(t) - A(s) - \rho(t-s)]. \quad (4)$$

In order to prove equation (4), note that the sequence A^{τ} is stopped at time τ, $\sigma(\tau)$, which is the maximum number of packet in the queue of a conserving link with a capacity of ρ and input A^{τ}.

C. SDN Switch Model

An SDN switch model is depicted in Fig. 2. This switch is a network element with one input A, one flow table control input C, and one output F such that $F = C(A(t))$, where $A(t)$ is the cumulative number of arrival by time t, $C(n)$ is the number of packet arrivals, which are flow controlled (e.g., using "Flow_Mod" operation) among the first n arrivals, and $F(t)$ is the cumulative packet departures by time t. In other words, the cumulative number of packet output by time t, is the cumulative number of packet arrival to the switch, which are "flow controlled" by the SDN-Controller by time t. For the case of OpenFlow, when the OpenFlow controller caches an operation inside the flow table of the switch, the consecutive packets, which match the flow table entry will not be forwarded to the OpenFlow controller. The operation of OpenFlow protocol, should be considered in the definition of $C(n)$ function. For an ideal SDN switch, if $A \sim (\sigma, \rho)$ is upper constrained and $C \sim (\delta, \gamma)$ is also upper constrained, then $F \sim (\gamma\sigma + \delta, \gamma\rho)$ is also upper constrained. This lemma can be easily proved as follows:

Proof:

$$\begin{aligned} F(t) - F(s) &= C(A(t)) - C(A(s)) \\ &\leq \delta + \gamma(A(t) - A(s)) \\ &\leq \delta + \gamma(\sigma + \rho(t-s)) \\ &= \delta + \gamma\sigma + \gamma\rho(t-s) \end{aligned}$$

■

We are assuming that the SDN switch is in fact a constant server under arrival process $A \sim (\sigma, \rho)$, with a constant service rate of μ (positive integer). We define a busy period for an SDN switch, which starts at instance s in time and ends at t if the queue length of the SDN switch at instances of $(s-1)$ and t is empty, while the queue length of the switch during the said period is not empty. In other words, $q(s-1) = 0, a(s) > 0, q(r) > 0$ for $s \leq r < t$ and $q(t) = 0$, where $q(t)$ is the length of the queue inside the SDN switch at time slot t. According to this definition, the duration B of the busy period is $B = t - s$ time units.

D. Analysis of the SDN switch

In the first part of our analysis we will compute and present the queue length and delay bound of SDN switch. Consider the SDN switch model (Fig. 2) as a single server model with a constant service rate (μ) and furthermore let A be the cumulative packet arrival for this SDN switch. As defined before, $q(t)$ will be the length of the packet queue at time slot t. Therefore, according to the definition of $a(t)$, and the packet forwarding rate of the SDN switch (i.e., μ), we have:

$$q(t+1) = (q(t) + a(t+1) - \mu)^+ \text{ with } q(0) = 0. \quad (5)$$

The $(exp)^+$ means that the expression is evaluated when it is positive and is zero otherwise. By using induction on t we can prove that we have

$$q(t) = \max_{0 \leq s \leq t} \{A(t) - A(s) - \mu(t-s)\}. \quad (6)$$

As the first step of induction, we know that $q(0) = 0$, therefore: $q(1) = \max(0, q(0) + a(1) - \mu)$, which can be re-written as:

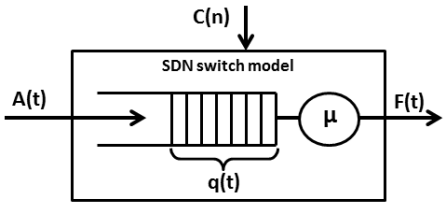

Fig. 2. A model of an SDN switch with interface to SDN controller.

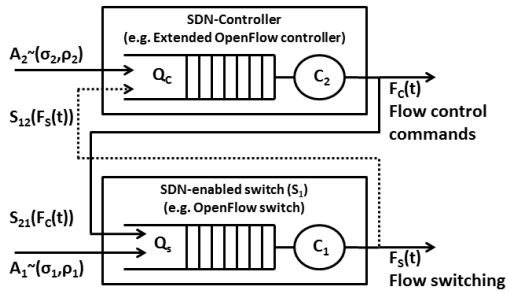

Fig. 3. A network model of SDN controller and SDN switches.

$\max_{0\le s\le 1}(A(1)-A(s)-\mu(1-s))$. Now suppose that (6) holds for t, it follows

Proof:

$$\begin{aligned} q(t+1) &= \max\{0, \\ &\max_{0\le s\le t}\{A(t)-A(s)-\mu(t-s)\}+a(t+1)-\mu\} \\ &= \max\{0, \\ &\max_{0\le s\le t}\{A(t+1)-A(s)-\mu(t+1-s)\}\} \\ &= \max_{0\le s\le t+1}\{A(t+1)-A(s)-\mu(t+1-s)\}. \end{aligned}$$

■

If we suppose that the packet arrival process (i.e., $A(t)$) is (σ,ρ)-upper constrained, and if $\mu \ge \rho$, then equation (6) implies that $q(t) \le \sigma \ \forall t \ge 0$. This is an upper bound independent of the service order. Using (6), we can compute the upper bound of the queue length inside the SDN switch. The output flow cumulative process of the SDN switch is:

$$F(t) = A(t) - q(t) = \min_{0\le s\le t}\{A(s)+\mu(t-s)\} \forall t \ge 0. \quad (7)$$

Based on the definition of "busy period", the SDN switch should have μ departures at each of the B times $\{s,\dots,t-1\}$. Also by definition we must have at least one packet in the queue at time $t-1$. Furthermore, at the time frame of $\{s,\dots,t-1\}$ at least $\mu B+1$ packets must arrive to sustain the busy period of the SDN switch. Since $A\sim(\sigma,\rho)$, we have at most $\sigma+\rho B$ packet arrivals during busy period B. This will help us to come up with a closed formula to express the busy period of the SDN switch using the constraints of the arrival process and its service rate. In fact we have $\mu B+1 \le \sigma+\rho B$, which can be simply re-written as (note that we present B in integer format):

$$B \le \left\lfloor \frac{\sigma-1}{\mu-\rho} \right\rfloor. \quad (8)$$

Finding the upper bound of the busy period, we can simply express the delay bound of the packets, which are residing inside the queue. We define that delay of a packet as the time instance the packet departs the SDN switch minus the time instance that it has arrived (and potentially forwarded to the SDN controller). The delay of any packet is less than or equal to the length of the busy period. Therefore, the upper bound of the packet delay, independent of service discipline is equal to the B, which is derived in (8).

E. Analysis of an SDN controller

The complete ecosystem of an SDN deployment includes a number of SDN switches, which are centrally controlled by an SDN controller. We have experimentally demonstrated the interaction of extended OpenFlow controller with packet and optical networking equipment in our recent works [6], [7]. Here we present an analytical model, based on network calculus theory, which provides a closed form of upper bound of queue length inside the SDN controller and SDN switches. The interaction of an SDN controller (e.g., NOX) and SDN switches are depicted in Fig. 3. Note that due to the multiplexing rule we can easily consider the packet input of other SDN switches in addition to the SDN switch, which is depicted in Fig. 2. The cumulative arrival process A_2 represents the packet inputs from other SDN enabled switches, which are controlled by centralized controller. Similarly part of the flow control commands, which leave the SDN controller is forwarded to the SDN switch 1 (S_1), and the rest is forwarded to other SDN switches, which are deployed in the network and are controlled by SDN controller. The output packet stream of S_1 (i.e., $F_S(t)$) is divided to two parts. One is forwarded to the SDN controller (i.e., $S_{12}(F_S(t))$), which represent the packet flows, for which the SDN switch has no flow-entry in its flow table. The other, represents the packet flow, for which an existing flow entry table is available in the flow table of the switch. We are assuming that both cumulative arrival processes (i.e., A_1, A_2) are upper constrained and the service rate of the queues in the SDN switch and controller are C_1 and C_2 respectively. The flow controlling functions $S_{12}\sim(\delta_{12},\gamma_{12})$ and $S_{21}\sim(\delta_{21},\gamma_{21})$ are upper constrained.

We are interested to find a closed form for the queue length of the SDN switch and controller. Let $\tilde{A}_1$ and $\tilde{A}_2$ be the overall arrival process of the SDN switch and SDN controller and $F_C(t)$, $F_S(t)$ be the respective output processes. We have

$$\tilde{A}_1(t) = A_1(t) + S_{21}(F_C(t)), \quad (9)$$

$$\tilde{A}_2(t) = A_2(t) + S_{12}(F_S(t)). \quad (10)$$

Furthermore, let F_C^τ and F_S^τ be the stopped sequence of F_C and F_S at time τ. It follows that for "any" α_1, $F_S^\tau \sim (\sigma_1(\tau),\alpha_1)$.

$$\sigma_1(\tau) = \max_{0\le t\le\tau}\max_{0\le s\le t}[F_S(t)-F_S(s)-\alpha_1(t-s)]. \quad (11)$$

Similarly, for "any" α_2, $F_C^\tau \sim (\sigma_2(\tau),\alpha_2)$.

$$\sigma_2(\tau) = \max_{0\le t\le\tau}\max_{0\le s\le t}[F_C(t)-F_C(s)-\alpha_2(t-s)]. \quad (12)$$

Assuming that $\gamma_{12}\gamma_{21}<1$, we have: $\alpha_1 = \frac{\rho_1+\gamma_{21}\rho_2}{1-\gamma_{12}\gamma_{21}}$ and $\alpha_2 = \frac{\rho_2+\gamma_{12}\rho_1}{1-\gamma_{12}\gamma_{21}}$. Considering the SDN switch model and multiplexing rule, we have:

$$F_S^\tau \sim (\sigma_1+\gamma_{21}\sigma_2(\tau)+\delta_{21}, \rho_1+\gamma_{21}\alpha_2), \quad (13)$$

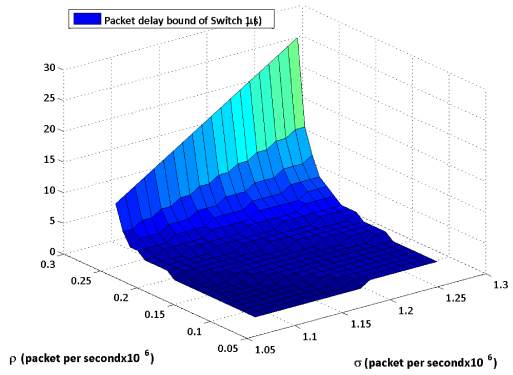

Fig. 4. Packet delay bound of Switch 1 vs. $A \sim (\sigma, \rho)$.

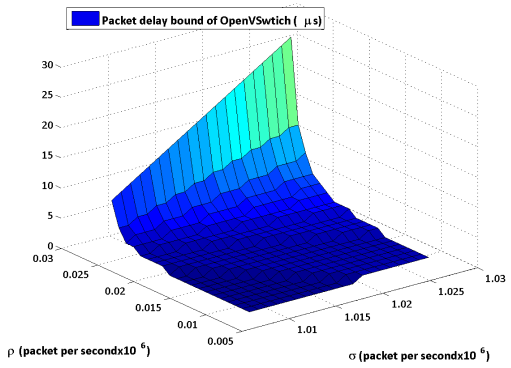

Fig. 5. Packet delay bound of Open vSwitch vs. $A \sim (\sigma, \rho)$.

Since the value of α_1 and α_2 are known, we can claim that $F_S^\tau \sim (\sigma_1+\gamma_{21}\sigma_2(\tau)+\delta_{21}, \alpha_1)$ is upper constrained. It follows that

$$\sigma_1(\tau) \leq \sigma_1 + \gamma_{21}\sigma_2(\tau) + \delta_{21}. \quad (14)$$

Using similar argument, we can characterize F_C^τ as follows:

$$\sigma_2(\tau) \leq \sigma_2 + \gamma_{12}\sigma_1(\tau) + \delta_{12}. \quad (15)$$

solving the above system of inequalities results in $\sigma_1(\tau) \leq \tilde{\sigma_1}$ and $\sigma_2(\tau) \leq \tilde{\sigma_2}$, where

$$\tilde{\sigma_1} = \frac{\sigma_1 + \gamma_{21}\sigma_2 + \gamma_{21}\delta_{12} + \delta_{21}}{1 - \gamma_{12}\gamma_{21}},$$

$$\tilde{\sigma_2} = \frac{\sigma_2 + \gamma_{12}\sigma_1 + \gamma_{12}\delta_{21} + \delta_{12}}{1 - \gamma_{12}\gamma_{21}}.$$

Note that we have just shown that these bounds are independent of τ. Therefore, F_C and F_S are both upper constrained (i.e., $F_S \sim (\tilde{\sigma_1}, \alpha_1)$ and $F_C \sim (\tilde{\sigma_2}, \alpha_2)$). This in turn implies that $\tilde{A_1}$ and $\tilde{A_2}$ are also upper constrained (i.e., $\tilde{A_1} \sim (\sigma_1+\gamma_{21}\tilde{\sigma_2}+\delta_{21}, \alpha_1)$ and $\tilde{A_2} \sim (\sigma_2+\gamma_{12}\tilde{\sigma_1}+\delta_{12}, \alpha_2)$). If $\alpha_1 \leq C_1$ then the queue length of the buffer inside the SDN switch (i.e., Q_S) is bounded by $\sigma_1 + \gamma_{21}\tilde{\sigma_2} + \delta_{21}$. Similarly, if $\alpha_2 \leq C_2$, then the queue length of the buffer (i.e., Q_C) inside the SDN controller is bounded by $\sigma_2 + \gamma_{12}\tilde{\sigma_1} + \delta_{12}$. To summarize:

$$Q_S \leq \sigma_1 + \gamma_{21}\tilde{\sigma_2} + \delta_{21}, \quad (16)$$

$$Q_C \leq \sigma_2 + \gamma_{12}\tilde{\sigma_1} + \delta_{12}. \quad (17)$$

It worth mentioning that our final results, which are the upper bounds of the buffer sizes in the switch and controller are independent of τ and mainly depend on upper bounds of the arrival processes and those of the SDN switch and SDN controller. Therefore, by characterizing the behavior of the arrival processes and operating regime of the SDN controller (through monitoring, or simulation studies), the upper bounds and buffer requirements of the SDN controller and switches can be easily computed.

IV. Evaluation Results

Using the mentioned analytical frameworks and presented outcomes, we report the upper bound of packet processing delay of two variations of OpenFlow switches and the upper bound of SDN controller buffer. The former gives a worst case estimation of packet processing delay of SDN switches, in contrast to the steady state (i.e., equilibrium) evaluation of classic queuing theory approaches. The latter is an interesting result for buffer sizing and performance evaluation of SDN controllers. In fact using the analytical results, the developers can precisely estimate the buffer requirement of controllers, given the specification of various arrival processes as described in Section III-E.

Fig. 4 and Fig. 5 depict the upper bound of the packet processing delay of two OpenFlow swithces, which are implemented in hardware and software respectively. We borrow the specification of these two switches from [13]. The first switch has an average packet processing performance of 0.286 million packets per second (mpps) and the second one, which is a software implementation (i.e., Open vSwitch) has an average processing performance of 0.03 mpps [13]. As shown in these results, the average sustainable arrival rate of the input packets to the soft-switch, should be kept one order of magnitude less than the same parameter of the hardware implementation (e.g., Switch 1 [13]) in order to achieve the similar performance in terms of packet processing delay.

Fig. 6 is one of the potential upper bounds, which yields the required buffer space in the SDN controller. Using the recent reports [23], we selected Beacon OpenFlow controller with an average performance of 1.75 million flow operations (e.g., Flow_Mod) per seconds [23]. Assuming the average arrival rate of 0.2 mpps for S1, 0.5 mpps as the aggregated arrival of other switches (see Fig. 3), the buffer requirement of this controller is shown in Fig. 6. We assumed that $1/100$ of the arrived packets will be forwarded to the controller (i.e., δ_{12}) with a burstiness parameter (i.e., γ_{12}) of 0.2 mpps. Given these parameters, the required buffer size of SDN controller amounts to 0.73 million packets in worst case scenario, which is about 1.02 GB if we consider an average packet size of 1500 bytes. This result helps the designers provision the required buffer space (i.e., buffer sizing) based on the operating regime of the switch in terms of average sustainable arrival rate and burstiness of the input traffic of the deployed OpenFlow switches along with the traffic specification of feedback and feed forward paths.

V. Conclusions and Future Research

In spite of benchmark tools (like OFlops and Cbench) and some limited simulation models, there are very few research activities to analytically evaluate the performance of an SDN

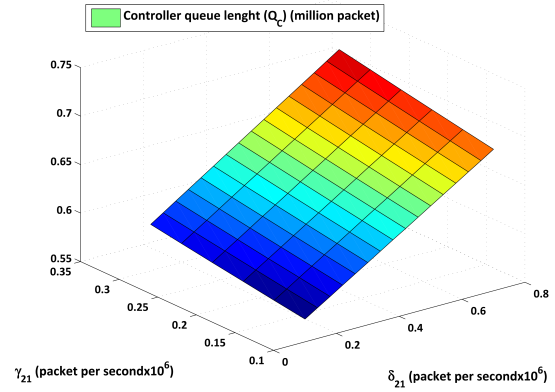

Fig. 6. Upper bound of SDN controller buffer vs $S_{21}(F_C(t))$.

deployment. In this work we exploited the capabilities of network calculus framework to model the behavior of an SDN switch and SDN controller. Focusing on bounds and worst case scenario, network calculus compliments the classical queuing theory. The latter concerns about the average quantities in equilibrium at the steady state, while the former focuses on boundary conditions. Our SDN switch model captured the closed form of the packet delay and buffer length inside the SDN switch according to the parameters of the cumulative arrival process. Furthermore, an analytical model of the interaction between SDN switch(es) and SDN controller were analysed. Given the parameters of the cumulative arriving processes and the flow control functionality of the SDN controller, the network architect or designer is able to compute an upper bound estimate of packet delay and buffer requirement of SDN switches. Using the recent measurement, we demonstrated the packet processing delay of two variants of OpenFlow switches along with the buffer requirement of an OpenFlow controller (i.e., Beacon). In addition to deterministic network calculus, stochastic network calculus is another interesting branch of network calculus, which can be also utilized for analytical modeling of other aspect of SDN. Utilizing the deterministic and stochastic network calculus models along with simulation and monitoring studies to identify the envelop of arrival and service curve of SDN deployments are among the list of our future studies.

REFERENCES

[1] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: a distributed control platform for large-scale production networks," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, 2010, pp. 1–6.

[2] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4d approach to network control and management," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 41–54, Oct. 2005.

[3] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, "Design and implementation of a routing control platform," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, ser. NSDI'05, Berkeley, CA, USA, 2005, pp. 15–28.

[4] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," in *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '07, New York, NY, USA, 2007, pp. 1–12.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[6] S. Azodolmolky, R. Nejabati, E. Escalona, R. Jayakumar, N. Efstathiou, and D. Simeonidou, "Integrated openflow–gmpls control plane: an overlay model for software defined packet over optical networks," *Opt. Express*, vol. 19, no. 26, pp. B421–B428, Dec 2011.

[7] M. Channegowda, R. Nejabati, M. R. Fard, S. Peng, N. Amaya, G. Zervas, D. Simeonidou, R. Vilalta, R. Casellas, R. Martínez, R. M. noz, L. Liu, T. Tsuritani, I. Morita, A. Autenrieth, J. Elbers, P. Kostecki, and P. Kaczmarek, "Experimental demonstration of an openflow based software-defined optical network employing packet, fixed and flexible dwdm grid technologies on an international multi-domain testbed," *Opt. Express*, vol. 21, no. 5, pp. 5487–5498, Mar 2013.

[8] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia, "Modeling and performance evaluation of an openflow architecture," in *Teletraffic Congress (ITC), 2011*, Sept., pp. 1–7.

[9] Y. Luo, P. Cascon, E. Murray, and J. Ortega, "Accelerating openflow switching with network processors," in *Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ser. ANCS '09, New York, USA, 2009, pp. 70–71.

[10] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "Openflow switching: Data plane performance," in *Communications (ICC), 2010 IEEE International Conference on*, May, pp. 1–5.

[11] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: scaling flow management for high-performance networks," in *Proceedings of the ACM SIGCOMM 2011 conference*, ser. SIGCOMM '11, New York, NY, USA, 2011, pp. 254–265.

[12] R. Pries, M. Jarschel, and S. Goll, "On the usability of openflow in data center environments," in *Communications (ICC), 2012 IEEE International Conference on*, June, pp. 5533–5537.

[13] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. Moore, "Oflops: An open framework for openflow switch evaluation," in *Passive and Active Measurement*, ser. Lecture Notes in Computer Science, N. Taft and F. Ricciato, Eds. Springer, 2012, vol. 7192, pp. 85–95.

[14] R. Sherwood and K.-K. Yap. Cbench controller benchmarker. [Online]. Available: http://www.openflow.org/wk/index.php/Oflops (last accessd on 12 March 2013)

[15] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Berkeley, CA, USA, 2012.

[16] M. Jarschel, F. Lehrieder, Z. Magyari, and R. Pries, "A flexible openflow-controller benchmark," in *Software Defined Networking (EWSDN), 2012 European Workshop on*, Oct., pp. 48–53.

[17] L. Kleinrock, *Queueing Systems*. Wiley Interscience, 1975, vol. I: Theory, (Published in Russian, 1979. Published in Japanese, 1979. Published in Hungarian, 1979. Published in Italian 1992.).

[18] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Springer-Verlag, 2001.

[19] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic (extended version)," *IEEE/ACM Trans. Netw.*, vol. 2, no. 1, pp. 1–15, Feb. 1994.

[20] Y. Jiang, "Stochastic network calculus for performance analysis of internet networks - an overview and outlook," in *Computing, Networking and Communications (ICNC), 2012 International Conference on*, 30 2012-Feb. 2, pp. 638–644.

[21] R. Cruz, "A calculus for network delay. i. network elements in isolation," *Information Theory, IEEE Transactions on*, vol. 37, no. 1, pp. 114–131, Jan 1991.

[22] ——, "A calculus for network delay. ii. network analysis," *Information Theory, IEEE Transactions on*, vol. 37, no. 1, pp. 132–141, Jan 1991.

[23] Openflow controller performance comparison. [Online]. Available: http://www.openflow.org/wk/index.php/Controller_Performance_Comparisons (last access 29 March 2013).