Deterministic Quality of Service Guarantee for Dynamic Service Chaining in Software Defined Networking

Yu-Jia Chen, Li-Chun Wang, *Fellow, IEEE*, Feng-Yi Lin, and Bao-Shuh Paul Lin

***Abstract*—In this paper, we present a systemic approach to provide deterministic delay guarantee for dynamic service chaining in software defined networking (SDN). The delay performance of service chaining in SDN is affected by signaling message exchange in control plane and packet transmissions in data plane, respectively. First, we develop an analytical method to characterize the delay performance of control plane when handling traffic with different priorities according to network calculus (NC) and queuing theory. Second, taking into account the estimated delay in control plane, we propose a novel service traversal mechanism to calculate the optimal traversal path for the service chain. We demonstrate that NC delay analysis can provide deterministic quality of service (QoS)-guaranteed service chaining for any specified delay requirements, whereas theoretical queueing delay analysis can only provide statistical QoS guarantee. In summary, the proposed NC delay analysis can help to understand the network design for a future delay sensitive Internet in which deterministic latency must be guaranteed.**

***Index Terms*—Service chaining, software defined networking, quality of service.**

I. Introduction

SOFTWARE defined networking (SDN) is shifting the paradigm of information service networks to automate resource provision with more flexibility and to optimize network performance more effectively and without geographical limitation. To obtain such advantages, SDN separates data and control planes by utilizing a centralized controller to manage the dynamic behavior of the entire network.

Service chaining is one of the key features for SDN. With this feature, SDN can dynamically connect virtual machines (VMs) with various functions to establish new services based on user requirements and network conditions [1]. For example, an advanced security service chaining can request the firewall and intrusion detection functions to inspect data sequentially, whereas a regular security service chaining process can only use the firewall server. Versatile service chaining processes have the great potential for creating new cloud services [2].

Manuscript received March 18, 2017; revised July 4, 2017, September 8, 2017, and September 19, 2017; accepted September 27, 2017. Date of publication September 29, 2017; date of current version December 8, 2017. This work was supported by the Ministry of Science and Technology, Taiwan, under Contract MOST 106-2221-E-009-027. The associate editor coordinating the review of this paper and approving it for publication was W. Kellerer. *(Corresponding author: Li-Chun Wang.)*

The authors are with the Department of Electrical and Computer Engineering, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: lichun@g2.nctu.edu.tw).

Digital Object Identifier 10.1109/TNSM.2017.2758328

The key to the success of dynamic service chaining in SDN is to satisfy the quality of service (QoS) requirements for each individual customer. To achieve this goal, three challenges must be overcome:

- Accurate VM workload estimation [3];
- Flow-based QoS evaluation for multiple data flows sharing the same physical links and nodes [4];
- Adaptation for rapid network topology changes [5].

The aforementioned challenges in providing QoS guarantees require joint modeling and analysis of both control and data plane. To our best knowledge, how to provide end-to-end deterministic delay grantee in service chaining simultaneously taking into account of both control and data planes is still an open issue.

In this paper, we aim to propose a systematic design methodology to provide QoS-guaranteed service chaining in SDN, considering the effects of both signaling message exchange in the control plane and packet transmissions in the data plane. The importance of including delay performance in the control plane lies in the fact that unexpected bursts or collapsed VM-based network functions may request the control plane to recreate a service chaining process for deciding a new service traversal. We suggest an analytical service chaining performance model for SDN meeting deterministic QoS requirements. The proposed systematic analysis methodology contains the following two key elements.

- *Delay Performance Evaluation in SDN Control Plane:* We apply queuing theory and network calculus (NC) to calculate the delay performance of the SDN controller in supporting various traffic types with different priorities. Queuing theory and NC can estimate the average delay and the deterministic delay bound of service chaining, respectively. The derived average and deterministic delay performances can be used to specify the delay constraints of service chain traversal in the data plane of SDN.
- *Path determination with Delay Constraints in SDN Data Plane:* We develop an optimal path determination algorithm for service chaining with minimum workload of network functions subject to the delay constraint of service chain traversal.

With the designed systematic methodology, we can explicitly determine the service chaining sequence and its routing path under different traffic loads and network sizes. Our experiments show that the proposed methodology can result in

service chaining satisfying the delay requirements of various users in a deterministic manner. We observe that the updating period of traversed paths in the control plane is a critical system parameter for service chaining with deterministic QoS. This work provides not only a delay performance evaluation approach, but also the design solution of achieving personalized QoS for network service providers.

The rest of the paper is organized as follows. Section II introduces the background of service chaining, SDN, and NC. We discuss the related works for service chaining in Section III. System model and problem formulation are given in Section IV. We analyze the delay performance of the SDN control plane in Section V. We discuss an optimal path determination problem for service chaining in Section VI, and present the system flow and framework for providing deterministic QoS-guaranteed service chaining in Section VII. Section VIII shows the experimental results for the delay performance of the service chaining with the proposed methodology. Finally, concluding remarks are given in Section IX.

II. Background

In this section, we introduce the background of service chaining, SDN, and NC.

A. Service Chaining

Service chaining is the deployment process of building a sequence of individual service functions to perform a complex task. For example, network service providers require a deployment model for advanced security services, such as intrusion detection and prevention systems, firewalls, content filters and optimization mechanisms. These security devices will be placed at a few choke points (e.g., core routers), and rely on routing to the desired network function deployment to enforce traffic inspection [6]. In that way, if a data flow comes from an untrusted source, it must traverse certain network service devices according to network policies, which is called rule-based forwarding [7]. Moreover, a user can specify which network service device that his/her packets should traverse. The network service provider thus forces the data flow through the desired sequences of network service devices. With rapid advances of server virtualization, network service is implemented by distributed VMs dedicated to one or a set of network service functions [8]. These VM-based network services can improve the scalability and flexibility of service chaining.

B. OpenFlow Software Defined Networking (SDN)

In SDN, network devices play a simple role of packet forwarding, while the intelligence or the control logics are implemented at the controller [9]. Therefore, SDN can increase the flexibility of network deployment, and the programmability as well as the reliability of network control functions and devices. To implement the SDN paradigm, OpenFlow is an open routing standard that enables researchers to realize SDN networks together with the current network. Fig. 1 illustrates an OpenFlow SDN, containing an OpenFlow controller

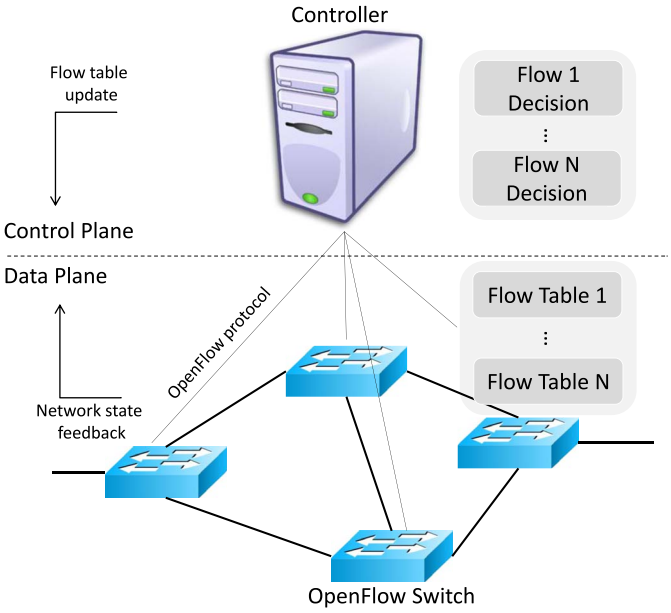

Fig. 1. System architecture of OpenFlow SDN.

to make flow decisions for OpenFlow switches. OpenFlow SDN separates the control plane (routing decision) from the data plane (packet forwarding), both of which are connected through the OpenFlow protocol. OpenFlow controller manages primitive network functions by programming the flow table of an OpenFlow switch [10]. According to the information in the flow table, each OpenFlow switch can forward incoming packets. The flow table at the OpenFlow switch is to record flow entries for the corresponding action commands [11], which can be dynamically added and removed based on the command from the OpenFlow controller. Hence, the network topology and the status of the entire network can be monitored by the OpenFlow controller.

C. Queueing Theory and Network Calculus (NC)

Both queueing theory and NC can provide delay performance modeling in SDN, but they are substantially different in analyzing the process of packet arrivals and services. In queuing theory based delay models, it is assumed that the inter-arrival time of a packet is exponentially distributed [12]. Based on this assumption, the expected service time and the delay distribution can be analyzed. Hence, queuing theory analysis can be applied for statistical delay guarantee of the network, such as ensuring that 99% of the traffic has a delay less than a certain threshold. However, it is extremely difficult to obtain the closed form result of delay distributions in general queueing theory models [13].

In contrast to queueing theory, NC places the emphasis on the quantitative level of packet arrival and service process models because packet inter-arrival time and service time are generally not exponentially distributed [14]. Specifically, NC handles the arrival and service process with boundary conditions rather than the equilibrium state of the network. In NC, data flows are characterized by traffic statistics and the inputs of packet networks are constrained by the use of traffic regulation schemes [15]. Based on this mathematical framework,

one can derive the deterministic performance bounds of delay, throughput and queue length in packet networks. In general, NC theory can be interpreted as a system theory for deterministic queueing systems. Therefore, NC is a powerful tool to analyze the network flow control problems from the worst case perspective and can obtain the delay bound for network performance evaluation [16].

Next, we give some basic definitions and notations for NC analytical approach. Consider a given data flow in a queueing system. The input and the output functions describe the arrival and the departure behaviors of the data flow in a cumulative perspective, respectively.

Definition 1 (Input Function): A cumulative input function $R(t)$ denotes the total number of packet arrivals in time slot 1, 2,..., t. $R(t)$ is a wide-sense increasing function in interval $[0, t]$, i.e., $R(0) = 0$ and $R(t_1) \leq R(t_2)$ if $t_1 \leq t_2$.

Definition 2 (Output Function): For the input function $R(t)$, a cumulative output function $R^*(t)$ denotes the total number of packet departures in time slot 1, 2,..., t. R^* is also a wide-sense increasing function in interval $[0, t]$.

Furthermore, NC considers the upper bound of the input function and the lower bound of the output function regarding the arrival and the service received by the data flow, respectively.

Definition 3 (Arrival Curve): Given a wide-sense increasing function α, the input function $R(t)$ has α as an arrival curve, or $R(t)$ is constrained by α, if and only if $R(t) - R(s) \leq \alpha(t - s)$ for all $s \leq t$.

Definition 4 (Service Curve): Consider a system S with input and output function R and R^*, respectively. We say that S offers a service curve β if and only if β is a wide-sense increasing function with $\beta(0) = 0$ and $R^*(t) \geq \inf_{s \leq t} [R(s) + \beta(t - s)]$.

Based on the arrival the service curves, we can calculate the delay bound of the data flow, which represents the worst-case response time of a packet.

Definition 5 (Delay Bound): For a flow $R(t)$ constrained by an arrival curve α and a system S offering a service curve β, when $R(t)$ traverses S for all t, the delay bound $D(t)$ must satisfy

$$D(t) \leq h(\alpha, \beta), \tag{1}$$

where

$$h(\alpha, \beta) = \sup_{t \geq 0} \{\inf[d(t) \geq 0, \alpha(t) \leq \beta(t + d(t))]\} \tag{2}$$

and $d(t)$ is the delay performance of the system S at time t as shown in Fig. 2.

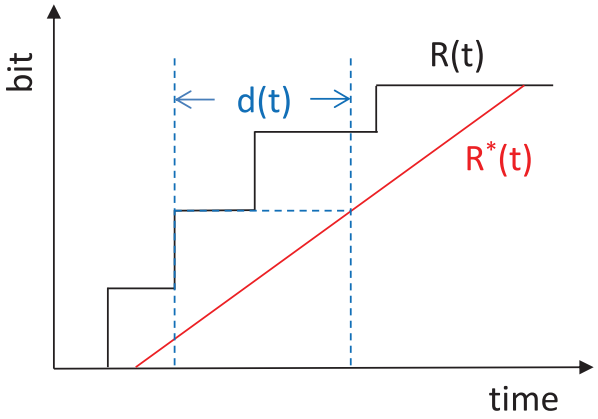

Fig. 2. Graphical representation of delay performance $d(t)$ with cumulative arrival process $R(t)$ and cumulative service process $R^*(t)$.

III. Related Work

In the literature, previous works regarding service chaining can be categorized into (1) service chain traversal in the data plane and (2) service chaining decision in the control plane.

- *Service Chain Traversal in Data Plane:* Switches in the data plane can be designed to support service chaining in cloud datacenters. Policy-aware switching was proposed to integrate a number of network functions into a pswitch that is deployed in datacenters [17]. This pswitch maintains a predefined policy rule table. Once receiving packets, the pswitch forces the packets to traverse a sequence of network functions and then forwards the packets to the next hop. It separates network functions from the data forwarding paths to improve flexibility and network operations. However, it has poor scalability and lacks resilience. A hybrid security architecture with hardware network functions and VM-based network functions coexisting in a datacenter was proposed in [18]. A label-based forwarding mechanism in VM-based network function agents can determine the next hop based on the label encapsulated in the packet header. These VM-based network functions are scalable for providing service chaining. Gember *et al.* [19] showed that distributed VM-based network functions cause inefficient manipulation and proposed a unified control plane to manage data flows through distributed network functions in service chaining. In [20], an optimization problem aiming to maximize throughput in service chaining under different topologies was investigated. Gushchin *et al.* [21] proposed a routing algorithm to maximize the number of service requests taking the constraints of switch memory and middle-box processing capacity into accounts.
- *Service Chaining Decision in Control Plane:* Despite active research on service chaining in the data plane, only a few works addressed the delay performance issues of service chaining decision in control plane. Tootoonchian *et al.* [22] designed a series of flow-based benchmarks and presented the OpenFlow controller performances using publicly-available OpenFlow controllers. It is emphasized that controller response time affects the delay performance of SDN significantly. The latency of communication between an OpenFlow switch and its controller was evaluated in [23]. Blenk *et al.* [24] compared the control plane latencies of different SDN hypervisor architectures and different network topologies. Similar to this work, a mathematical framework based on NC to characterize the behaviors of the OpenFlow controller

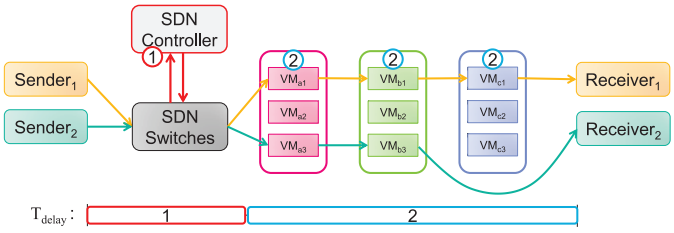

Fig. 3. System model of service chaining.

was presented in [25]. Based on the NC analytical framework, the upper bound of packet delay and the buffer requirement of the control plane can be calculated.

Although previous works have analyzed the delay performance of data plane and control plane in SDN, these works tackle the problems of delay evaluation and routing decision separately. Hence, there is still a gap between understanding the theoretical limits of delay performance and making optimal network decisions. In our previous work [26], we developed a security traversal mechanism to achieve traversal delay guarantee but only considered the impact of data plane. In this paper, we propose a generalized service chaining analytical model with the consideration of the delay performance of both control plane and data plane. The novelty of the proposed model is mainly contributed by formulating a constrained shortest path problem with SDN delay performance modeling so that we can provide the algorithm for path determination in service chaining. With the proposed analytical model, we can determine the service chain traversal for any deterministic delay requirement.

TABLE I
NOTATIONS IN THIS PAPER

Notations	Descriptions
r	Routing path
T_{Delay}	Delay guarantee
T_C	Service chaining decision delay
T_D	Service chain traversal delay
$T_{C\_upperbound}$	Upper bound of the service chaining decision delay
$T_{D\_constraint}$	Constraint for T_D
$\mathrm{G}(M, L)$	Network with M network functions and L links
R_{st}	Set of paths from source node s to destination node t
$f_C(r)$	Load function of path r
$f_D(r)$	Delay function of path r
c_k	Load of the k^{th} network function
$m(r)$	Number of network functions on path r
d_k	Delay metric for the k^{th} network function
λ_c	Arrival rate in queueing theory
μ_c	Service rate in queueing theory
L_c	Expected number of requests in the controller
W_c	Expected waiting time in the controller
R_c	Input function of the controller in network calculus
R_c^*	Output function of the controller in network calculus
α	Arrival curve of R_c
β	Service curve of R_c^*
D_c	Delay bound of the controller
$l_{\max}$	Maximum data size of the flow
b	Maximum buffer size of the controller
$f_G(r)$	Network congestion measure on path r
T_{ij}	Total measured traffic amount on link (i, j)
B_{ij}	Max achievable bandwidth on link (i, j)
λ	Lagrangian multiplier
$f_\lambda(r)$	Aggregated cost of path r
$\mathbf{c}$	Vector of network function loading f_C
$\mathbf{d}$	Vector of delay coefficients f_D
m	Total number of network functions
n	Total number of links

IV. System Model and Problem Formulation

We now discuss the system model of the service chaining in SDN. The service chaining scenario considered in this paper consists of several distributed VM-based network functions and a centralized OpenFlow controller. To achieve delay guarantee T_{Delay}, both the service chaining decision delay T_C and the service chain traversal delay T_D should be considered as shown in Fig. 3. When data flows arrive at the OpenFlow switches, the OpenFlow controller decides the optimal setting of the service chaining, including the network function selection and the traversal path decision. This kind of operation delay is called the service chaining decision delay T_C. Then, the data flows traverse the corresponding VMs with the specified network functions. The total service chain traversal delay is denoted as T_D. As a result, the relation of the user's delay requirement and the service chaining delay is written as

$$T_{Delay} = T_C + T_D. \tag{3}$$

To achieve the deterministic delay requirement, we adopt the upper bound of the service chaining decision delay $T_{C\_upperbound}$ in determining the delay constraint on the data plane

$$T_{D\_constraint} = T_{Delay} - T_{C\_upperbound}, \tag{4}$$

where $T_{D\_constraint}$ will be the constraint for the optimal path determination problem of the service chaining.

We now formulate the path determination problem of service chaining. Consider a network represented by a directed simple graph $\mathrm{G}(M, L)$, where M is the set of all VM-based network functions and L is the set of all links. We denote an ordered pair outgoing from node i and incoming to node j by link (i, j). Let R_{st} be the set of all the paths from source node s to destination node t. For a path $r \in R_{st}$, we define the load function f_C and the delay function f_D as

$$f_C(r) \triangleq \max_{1 \le k \le m(r)} \{c_k\} \tag{5}$$

and

$$f_D(r) \triangleq \sum_{k=1}^{m(r)} d_k, \tag{6}$$

where c_k is the load of the k^{th} VM-based network function, $m(r)$ is the number of VM-based network functions on path r, and d_k is the delay metric for the k^{th} VM-based network function. The path determination problem in service chaining is to find a routing path r^* with the minimum VM-based network function loading f_C subject to the delay constraint $f_D(r^*) \le T_{D\_constraint}$. Table I summarizes the notations used in this paper.

V. Delay Performance Analysis of the Control Plane

Now we analyze the delay performance of the OpenFlow controller with priority traffic and non-priority traffic. We analyze the average delay performance and deterministic delay bound in the service chaining decision process. Both of the derived results are used to determine the delay constraints of service chain traversal.

A. Queueing Analysis for Average Delay Performance

We use the M/M/1 queueing model to characterize the delay performance of an OpenFlow controller. We consider the first-in-first-out (FIFO) policy. Let λ_c and μ_c be the arrival rate and service rate. For the M/M/1 queueing model, the expected number of requests in the system can be written as

$$L_c = \frac{\lambda_c}{\mu_c - \lambda_c}. \tag{7}$$

From Little's formula [27], we can obtain the expected waiting time W_c at the OpenFlow controller as

$$W_c = \frac{1}{(\mu_c - \lambda_c)}. \tag{8}$$

Next, we derive the delay performance in the case of arrival packets with two different priority levels. Let the high priority and the low priority data flows have a mean arrival rate λ_c^h and λ_c^l, respectively. We consider non-preemption priority queueing in this paper. In a non-preemption priority system, the high priority data flow is served earlier than the waiting low priority data flow in the queue, but the serving low priority data flow can complete its service upon the arrival of the high priority data flow. The traffic intensity for high priority and low priority traffic is $\rho_c^h \equiv \frac{\lambda_c^h}{\mu_c}$ and $\rho_c^l \equiv \frac{\lambda_c^l}{\mu_c}$, respectively. Referring to [28], we can obtain the expected number of high priority and low priority data flows as

$$L^h = \frac{\lambda_c^h(\rho_c^h + \rho_c^l)}{\mu_c - \lambda_c^h} + \frac{\lambda_c^h}{\mu_c} \tag{9}$$

and

$$L^l = \frac{\lambda_c^l(\rho_c^h + \rho_c^l)}{(\mu_c - \lambda_c^h)(1 - \rho_c^h - \rho_c^l)} + \frac{\lambda_c^l}{\mu_c}, \tag{10}$$

respectively. Based on the Little's formulas, the expected waiting time of the high priority and low priority data flows can be written as

$$W^h = \frac{L^h}{\lambda_c^h} = \frac{(\rho_c^h + \rho_c^l)}{\mu_c - \lambda_c^h} + \frac{1}{\mu_c} \tag{11}$$

and

$$W^l = \frac{L^l}{\lambda_c^l} = \frac{(\rho_c^h + \rho_c^l)}{(\mu_c - \lambda_c^h)(1 - \rho_c^h - \rho_c^l)} + \frac{1}{\mu_c}, \tag{12}$$

respectively.

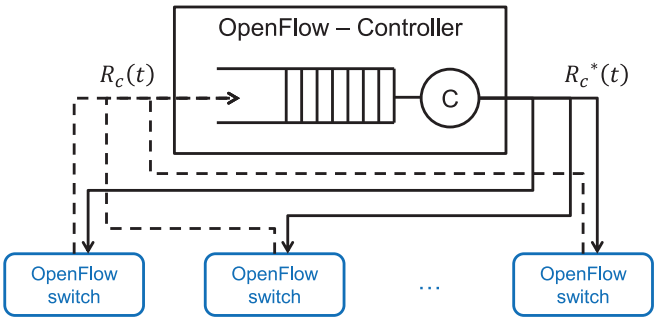

Fig. 4. Network calculus model for OpenFlow controller.

B. NC-Based Analysis for Delay Performance Bound

We now analyze the delay upper bound of an OpenFlow controller with non-priority traffic using NC. The considered NC model for an OpenFlow controller and OpenFlow switches is shown in Fig. 4. The cumulative arrival process $R_c(t)$ represents the packet inputs from OpenFlow switches which have no matched flow entry in their flow tables. The cumulative output process $R_c^*(t)$ represents the flow control command to the requested OpenFlow switches for forwarding information.

In the following, we derive the closed form for the delay upper bound of the OpenFlow controller. Let α and β be the arrival curve of $R_c(t)$ and service curve of $R_c^*(t)$, respectively. Then the delay bound $D_c(t)$ of T_C for all t satisfies:

$$D_c(t) \le h(\alpha, \beta), \tag{13}$$

where

$$h(\alpha, \beta) = \sup_{t \ge 0}\{\inf[d(t) \ge 0, \alpha(t) \le \beta(t + d(t))]\} \tag{14}$$

and

$$d(t) = \inf\{\tau \ge 0 | R(t) \le R^*(t + \tau)\}. \tag{15}$$

Consider an OpenFlow controller with service curve c serving high priority data flow H and low priority data flow L. Let the arrival curve of H be α^h. Then we show the derivation of delay performance bound analysis for both traffic priority classes.

- Since the high priority flow is guaranteed by a service rate with curve c, we can analyze the delay bound $D_c^h(t)$ in two cases. In the first case, we consider that the high priority flow is served in the OpenFlow controller when a high priority flow arrives. That is, the incoming data flows and the existing data flows have the same priority. Because it can be reduced to the non-priority traffic case with arrival curve α^h and service curve c, we obtain the delay bound

$$D_c^{h1}(t) \le h\left(\alpha^h, c\right), \tag{16}$$

where $h(\alpha^h, \beta)$ can be solved by substituting α^h and c into (2). Thus we have

$$D_c^{h1}(t) \le \sup_{t \ge 0}\left\{\inf\left[d(t) \ge 0, \alpha^h(t) \le c(t + d(t))\right]\right\}. \tag{17}$$

In the second case, we consider a low priority data flow is served in the OpenFlow controller when a high priority

flow arrives. In this case, the incoming data flow must wait for the low priority data flow in the controller. After this low priority data flows leave the controller, the high priority data flows can be served, which can be reduced to the first case. Thus, the delay bound consists of the service time of a low priority data flow and the delay bound of the first case.

$$D_c^{h2}(t) \leq \frac{l_{\max^L}}{c} + D_c^{h1}, \tag{18}$$

where $l_{\max}^L$ is the maximum data size of the low priority flow and D_c^{h1} is the delay bound for the first case.

- For the low priority traffic, the worst case is that it must wait until all the other low priority traffic leaves the system. Hence, the low priority data flows are guaranteed by a service rate of curve $c - \alpha^h$ and the delay bound is written as

$$D_c^l(t) \leq \frac{b}{c - \alpha^h}, \tag{19}$$

where b is the maximum buffer size of the system.

VI. Optimal Path Determination in Service Chaining

Now we show how to determine the optimal path of service chaining in the OpenFlow controller. Because a data flow traverses a set of VM-based network functions sequentially in the service chaining, the delay performance of service chaining is also affected by the VM-based network functions in the data plane. Hence, referring to the load function defined in (5), the path determination problem in service chaining can be defined to find a routing path with the minimum load of these network functions. Based on the aforementioned worst case delay performance of the OpenFlow controller, we can obtain the constraint $T_{D\_constraint}$ for service chain traversal delay as in (4). This constraint gives the maximum amount of time that can be spent in all the network functions on a service chaining path.

We model the path determination problem as a constrained shortest path (CSP) problem. The CSP problem can be formally stated as follows:

$$r^* = \arg\min_{r \in R_{st}} \{f_C(r) | f_D(r) \leq T_{D\_constraint}, f_G(r) = 0\}, \tag{20}$$

where $f_G(r) = \sum_{(i,j)\in r} g_{ij}$ is the network congestion measure for the traffic on path r. In practice, a link is congested if link utilization exceeds 70% [29]. Here we define g_{ij} as

$$g_{ij} = \begin{cases} \frac{T_{ij} - 0.7 \times B_{ij}}{T_{ij}}, & 0.7 \times B_{ij} < T_{ij} \\ 0, & 0.7 \times B_{ij} \geq T_{ij} \end{cases}, \tag{21}$$

where T_{ij} is the total measured traffic amount in bps and B_{ij} is the max achievable bandwidth in bps on link (i, j). With this network congestion constraint, we can avoid link overloading for a large number of paths with the same R_{st}.

In (20), we find the minimum load service chaining path r^* satisfying a specified delay requirement. Since the CSP problem is NP-complete [30], we apply the Lagrangian relaxation based aggregated cost (LARAC) algorithm [31] in our proposed scheme. The LARAC algorithm can solve the dual problem of the CSP problem. It follows that

$$\begin{aligned} &\text{maximize} \quad \min\{f_\lambda(r) - \lambda D_{\max} | r \in R_{st}\} \\ &\text{subject to} \quad \lambda \geq 0, \end{aligned} \tag{22}$$

where λ is the Lagrangian multiplier and

$$f_\lambda(r) = \sum_{k \in m} (c_k + \lambda d_k). \tag{23}$$

Note that $f_\lambda(r)$ represents the aggregated cost for path $r \in R_{st}$. The lower bound for the optimal solution of (20) can be obtained by finding $\{f_\lambda(r) | r \in R_{st}\}$ with Dijkstra's shortest path algorithm because

$$\min\{f_\lambda(r) - \lambda D_{\max} | r \in R_{st}\} = \{f_\lambda(r) | r \in R_{st}\} - \lambda D_{\max}. \tag{24}$$

Algorithm 1 shows the LARAC algorithm, where $\mathbf{c}$ is the vector of network function loading f_C, $\mathbf{d}$ is the vector of delay coefficients f_D, and $\mathbf{f}_\lambda$ is the vector of aggregated loading coefficients f_λ. The procedures of Algorithm 1 are described as follows.

1) The algorithm calculates the shortest path with network function loading, which is denoted by p_C. If a path meets the delay constraint $T_{D\_constraint}$, it is the optimal path with minimum loading. Otherwise, the algorithm stores the path as the best path that does not satisfy $T_{D\_constraint}$ and computes the shortest path with delay coefficients, which is denoted by p_D. If p_D still cannot meet the delay requirement, there is no solution to this problem.
2) With $\lambda = \frac{f_C(p_C) - f_C(p_D)}{f_D(p_D) - f_D(p_C)}$, we can find a new shortest path r with aggregated cost f_λ given in equation (23). If $f_\lambda(r) = f_\lambda(p_C)$, the optimal λ is obtained as proved in [32]. Otherwise, either p_D or p_C is replaced with r according to whether the delay constraint is satisfied or not.

The complexity of the LARAC algorithm is $O([m + n \log n]^2)$, where m and n are the total number of network functions and links, respectively [31].

VII. Providing Deterministic QoS-Guaranteed Service Chaining in SDN

Now we discuss how the delay analysis in Sections V and VI can be applied to provide deterministic QoS-guaranteed service chaining. According to (4), $T_{D\_constraint}$ can be obtained by subtracting the derived $T_{C\_upperbound}$ from a given delay requirement T_{Delay}. Next, we solve the shortest path determination problem for service chaining subject to the constraint $f_D(r) \leq T_{D\_constraint}$.

Figure 5 illustrates the system flow of the proposed deterministic QoS-guaranteed service chaining. First, a service chain is defined by the requested network functions and its delay requirement. Then the system dynamically monitors the traffic load of each link and the CPU usage of the VM-based network functions, which affect the delay performance in the control plane. After that, LARAC algorithm can be used to solve a path determination problem. If we cannot find a service chain satisfying the delay requirement, some VM-based network functions are added to reduce the delay performance.

Algorithm 1 LARAC Algorithm

Require: s, t, $\mathbf{c}$, $\mathbf{d}$, $\mathbf{f}_\lambda$, $T_{D\_constraint}$
Ensure: Find λ
  $p_C = Dijkstra(s, t, \mathbf{c})$
  if $f_D(p_C) \leq T_{D\_constraint}$ **then**
    return p_C
  else
    $p_D = Dijkstra(s, t, \mathbf{d})$
    if $f_D(p_D) > T_{D\_constraint}$ **then**
      return "there is no solution"
    else
      while *true* **do**
        $\lambda = \frac{f_C(p_C) - f_C(p_D)}{f_D(p_D) - f_D(p_C)}$
        $r = Dijkstra(s, t, \mathbf{f}_\lambda)$
        if $f_\lambda(r) = f_\lambda(p_C)$ **then**
          return p_D
        else if $f_D(r) \leq T_{D\_constraint}$ **then**
          $p_D = r$
        else
          $p_C = r$
        end if
      end while
    end if
  end if

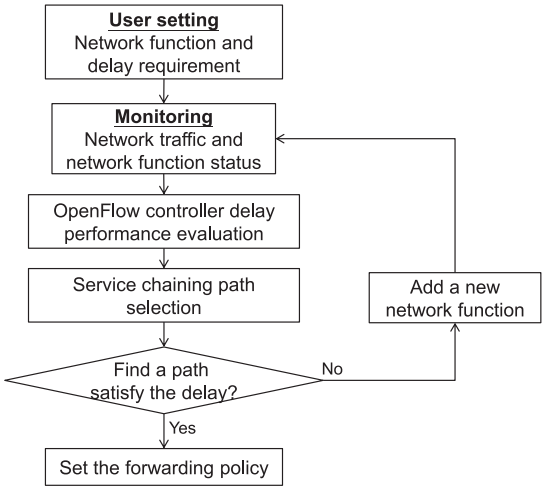

Fig. 5. System flow for providing deterministic QoS-guaranteed service chaining.

Finally, the OpenFlow controller updates the flow table of OpenFlow switches.

We now present the system framework and the corresponding process for providing deterministic QoS-guaranteed service chaining in OpenFlow networks. The system framework consists of four modules in the OpenFlow controller and switches, as shown in Fig. 6. The process between these modules can be classified into three stages: network condition information collecting, service chaining decision making, and traffic flow controlling. We then describe these stages as follows.

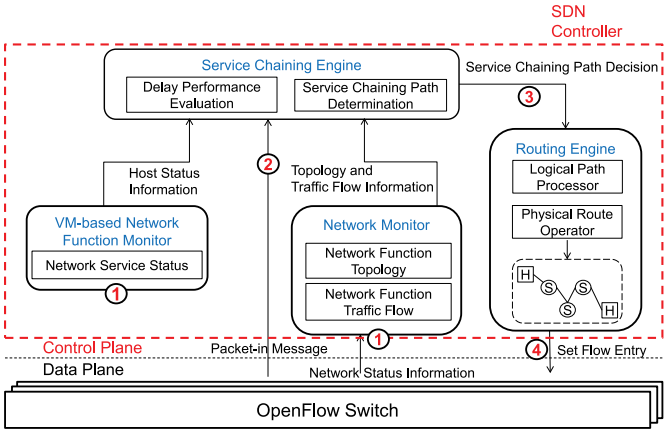

Fig. 6. System framework for providing deterministic QoS-guaranteed service chaining in OpenFlow networks.

TABLE II
AN EXAMPLE OF HOST-PORT-SWITCH TABLE

Host IP	Host MAC	Switch	Port number
10.0.0.1	66:d3:c2:48:e4:af	00:00:00:00:00:00:00:06	1
10.0.0.2	72:50:cf:b0:2e:42	00:00:00:00:00:00:00:07	1

TABLE III
AN EXAMPLE OF SWITCH-LINK MATRIX

	Switch 1	Switch 2	Switch 3	Switch 4
Switch 1	x	3	x	2
Switch 2	3	x	4	5
Switch 3	x	3	x	4
Switch 4	2	3	4	x

A. Network Condition Collecting

In the first stage, the status of VM-based network functions and network conditions are collected by the monitor module. The network function monitor records the CPU usage of the VM. In addition, the network condition monitor is responsible to record the topology information and the traffic flow via the OpenFlow switches. The network topology can be represented as a host-port-switch table and a switch-link matrix. The former is used to record how end hosts are connected to the OpenFlow switches, including the IP address and the MAC address of the end host, the OpenFlow switch ID, and the port number as shown in Table II. The switch-link matrix is used to record how the OpenFlow switches are connected each other as shown in Table III.

B. Service Chaining Decision Making

In this stage, the service chaining engine analyzes the delay performance and then determines the optimal service chaining path based on the network condition. According to the OpenFlow protocol, when receiving a packet of a new data flow, the switch forwards this new packet to the controller. Then, the service chaining engine checks the packet header to obtain the source/destination host addresses and evaluates the delay performance based on the network conditions. After that, the optimal routing path of service chaining is determined by solving a CSP problem subject to the network functions and delay requirements. Finally, the routing information is

updated. To ensure the delay guarantee, the service chaining engine periodically checks the delay performance of the service chain. The optimal routing path will be recalculated if the delay performance of the service chain violates the delay requirement.

C. Traffic Flow Controlling

In the final stage, the routing engine receives the optimal path decision from the service chaining engine. Then the logical path is translated to the explicit network routes. The path-to-route translator divides the path into multiple end-to-end network routes and computes the shortest route from one host/network function to another based on the current network topology. Finally, the route operator converts the shortest route to flow entries and updates to each OpenFlow switch on the route.

VIII. Experimental Results

To demonstrate the effectiveness of the proposed generalized service chaining analytical model, we conduct a series of experiments. We verify that the proposed NC-based delay performance evaluation can precisely estimate the upper bound of the controller delay performance for different controller loadings, network scales, and traffic generation rate. Hence, the deterministic QoS-guaranteed service chaining with multiple delay requirements can be achieved. We also discuss the tradeoff between QoS violation and controller loading.

Here we first detail the experimental environment and assumptions. We test the proposed service chaining over a standard Floodlight OpenFlow controller [33], which is a Java-based OpenFlow controller, open-source with Apache-licensed software, and is supported by a community of developers. Floodlight offers a systematic development architecture with various application programming interfaces (APIs). The proposed system is implemented in a real network environment as well as a virtualized network environment. For the real network environment, we use open-source software package OpenWRT [34] for OpenFlow v1.0 [35] with TP-Link TL-WR1043ND as the OpenFlow switch. We also set up the VMs as the end hosts and the network functions. For virtualized network environment, we use Mininet [36] to emulate the entire network topology. Mininet can simulate a network interconnected by end hosts, switches, and controllers. Both the real and the virtualized network environments are controlled by the Floodlight controller. We adopt the mesh topology aggregated by multiple OpenFlow switches with two VMs linked to each switch. We integrate Libvirt [37] API with the host operating system of each server node to obtain the CPU usage in the network functions. To monitor the status of VM-based network functions and network conditions, the monitor module is implemented in each server node and runs as a background process. The monitor module is responsible for reporting the CPU usage and transit traffic to the Floodlight controller. By using the Floodlight API, we can get notified about any changes on the switches, ports, and links. Hence, we can maintain the topology relevant information as described in Section VII-A.

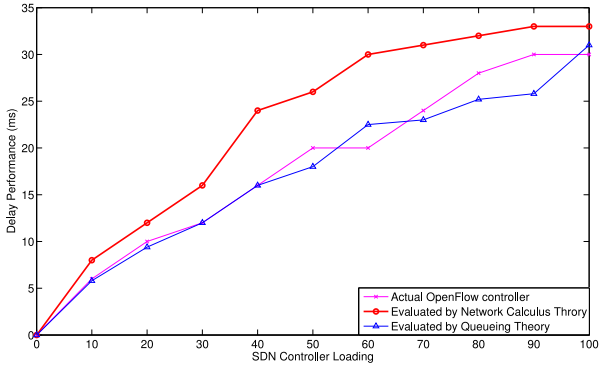

Fig. 7. Comparison of delay performance evaluation using network calculus theory and queueing theory.

A. Controller Delay Performance Evaluation Using Queueing Theory and NC

Figure 7 shows that the delay performance based on NC analysis can be upper bounded under different controller loadings. In contrast, queueing theory gives the average delays in equilibrium, which are smaller than the actual delay of an OpenFlow controller in six times out of ten. One can improve the accuracy of delay evaluation using queueing theory by deriving the delay cumulative distribution function and selecting the delay value that exceeds the 95th percentile or higher. Nonetheless, it is still an open issue to obtain the delay cumulative distribution function in service chaining since the packet arrival process and the service time distribution from different senders are generally non-homogeneous [38].

Figure 8 shows the delay performance of an OpenFlow controller with different numbers of VM and traffic generation rate. It is observed that the delay performance of the controller degrades severely in service chaining when a datacenter scales up. Furthermore, we observe that traffic generation rate affect the delay performance of the controller. A higher traffic generation rate causes a higher delay of the controller, especially at a large VM scale. It is shown that the delay of the controller with 75% traffic generation rate is nearly twice as that with 25% traffic generation rate. Thus, the delay in the control plane is very important, and cannot be ignored. In addition, our results indicate that NC-based delay analysis provides a tight upper bound delay for a small number of VM scales. Specifically, the upper bound delay is only 10% larger than the emulated result for the network scales under 60 VMs. It is worth mentioning that the controller delay can be up to 100 ms for the network scales under 120 VMs with 75% traffic generation rate. Since the end-to-end latency requirement for ultra-low latency services has to be under 100 ms in the future tactile Internet [39], it is implied that the controller delay performance is one of the primary factors that need to be considered. Our results can be extended to multi-controller networks in which each controller is responsible for its own network slice with 120 VMs and 75% traffic generation rate if the cost of inter-controller communications [40] is ignored.

Finally, we note that the possible delay from the monitor process should be considered in providing deterministic

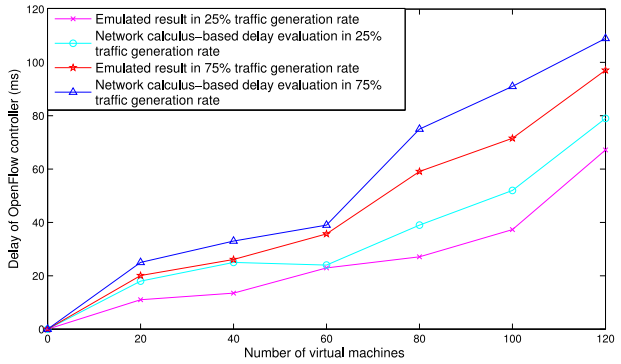

Fig. 8. Delay performance of OpenFlow controller with different VM scales and traffic generation rate.

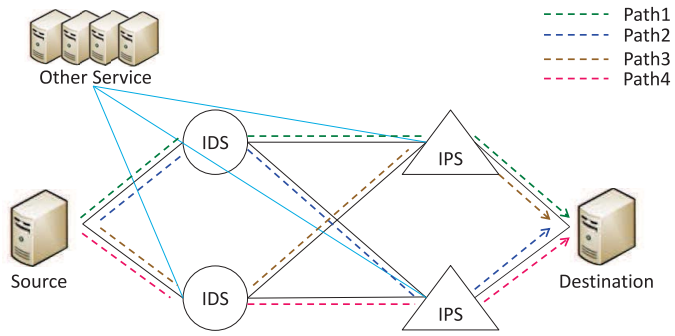

Fig. 9. Experiment scenario of security service chaining.

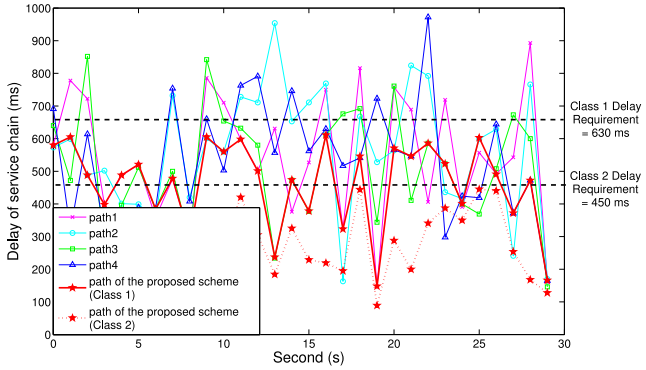

Fig. 10. Instantaneous delay performance of the service chaining paths with different delay requirements.

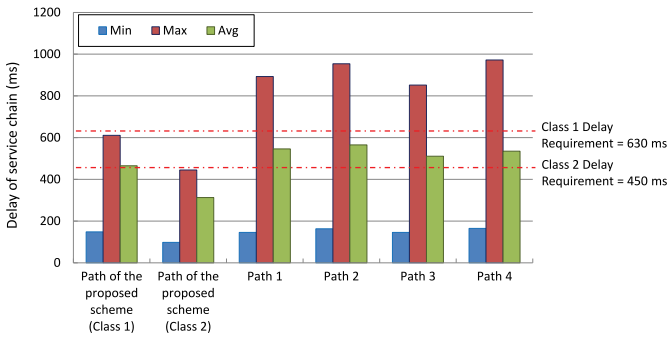

Fig. 11. Statistical delay performance of the service chaining paths with different delay requirements.

QoS-guaranteed service chaining. The delay from the monitor process is contributed by the interval processing time of the monitoring messages and the propagation time of OpenFlow messages. In our experiments, the interval processing time of the monitoring messages and the propagation time of OpenFlow messages are 1.1 ms and 4.8 ms on average, respectively. Since the OpenFlow messages are delayed by the link propagation and queuing in the input and output ports along the path, the propagation delay of control traffic can be reduced by placing the SDN controller in the most central node of the network [41].

B. Delay Performance for Multiple Delay Requirements and Different Re-Traversal Periods

Figure 9 illustrates our experiment scenario with multiple security service chaining paths from source to destination. We assume that the data sent by the source host needs to be checked by the intrusion detection system and the intrusion prevention system sequentially.

Figure 10 shows that the proposed scheme can provide delay guarantee with different delay requirements. The experimental scenario assumes that the delay requirement of class 1 users is 630 ms. At the 10th second, class 2 users specify that the delay requirement is 450 ms. As shown in Fig. 11, even though the average delay performance of all the paths can satisfy the delay requirement 630 ms, only the path of the proposed scheme can provide delay guarantee in the worst case for both class 1 and class 2 users.

Note that in our proposed scheme if we cannot find a path satisfying the delay requirement, a new VM will be added to balance the loading of network functions, thereby resulting in new paths with shorter delay. Then the OpenFlow controller recalculates the service chaining path to ensure the QoS. Clearly, the new VM should be the same network function as that with the maximum loading (i.e., bottleneck point) for all the paths and placed as near as possible to the previous hop of the bottleneck point. Similarly, taking into account of traffic routing, the VMs in service chaining should be placed so as to avoid bottleneck points and distribute workloads of the overloaded network functions.

Figure 12 and 13 show the delay performance of the proposed scheme with the four service chaining paths under re-traversal period equal to one unit time and five unit time, respectively. The re-traversal period is defined as the interval between two delay performance updates of the service chaining path. One can observe that the proposed service chaining scheme can always ensure the allowable delay if the OpenFlow controller periodically checks the delay performance of the service chaining path every one unit time (i.e., re-traversal period equal to one unit time). However, some violations occur for the re-traversal period equal to 5 unit time. As shown in Fig. 14, we cannot provide worst-case delay guarantee with a large re-traversal period. Table IV compares the number of delay violation and the number of path reconfigurations for different re-traversal periods within 60 unit time. However, a shorter re-traversal period will cause higher computing and

TABLE IV
COMPARISON OF THE DELAY VIOLATION FOR DIFFERENT RE-TRAVERSAL PERIODS WITHIN 60 UNIT TIME

	Delay violation times	Percentage of violation	Path reconfiguration times
Re-traversal period=1	0	0	22
Re-traversal period=5	15	0.25	4

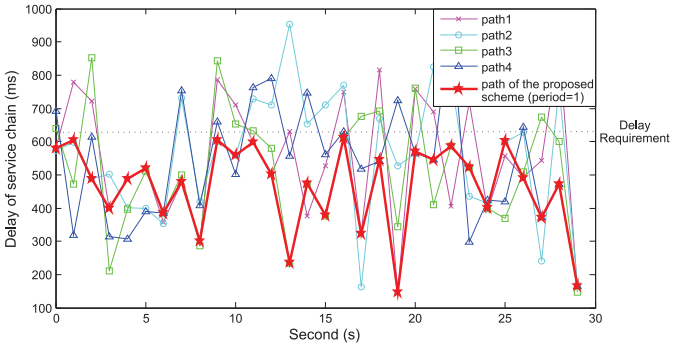

Fig. 12. Instantaneous delay performance of the four service chaining paths and the proposed dynamic service chaining scheme with re-traversal period equal to one unit time.

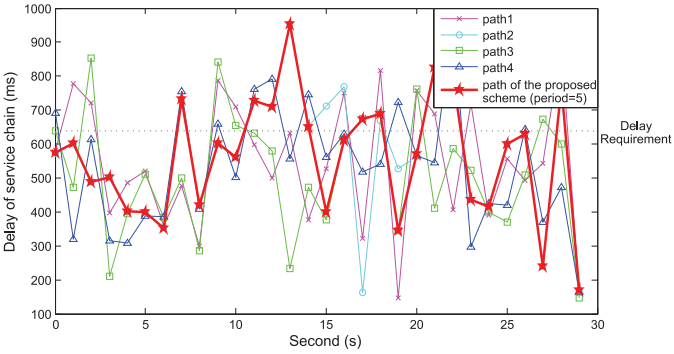

Fig. 13. Instantaneous delay performance of the four service chaining paths and the proposed dynamic service chaining scheme with re-traversal period equal to 5 unit time.

communication overhead in the control plane. Therefore, it is crucial to take the delay requirement and the network conditions into account when determining a re-traversal period. We note that a shorter re-traversal period implies a larger reconfiguration cost resulted from path computation and path updates. The computation overhead of the proposed algorithm is relatively small compared to the delay requirement. In our experiment, it costs only 1.4 ms on average to make the path decision. Note that the computation overhead of the proposed algorithm depends on the number of hops of the shortest path between the sender and the receiver. Fortunately, it has been shown that the number of hops of the shortest path between the sender and the receiver is small even when the number of nodes is large for the random and power-law out degree networks [31]. Moreover, path updates will yield packet loss if the update process cannot be synchronized across all the switches on the path [42]. In recent years, an accurate time-based approach that simultaneously performs multiple changes at different switches has proven to be viable for frequent network updates by using a timestamp field in the switch [43].

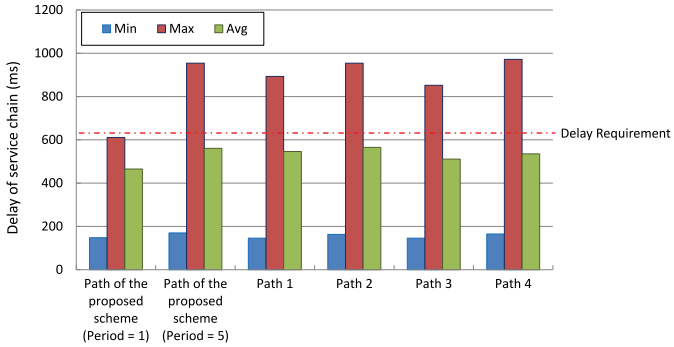

Fig. 14. Statistical delay performance of the service chaining paths with different re-traversal periods.

From the aspect of energy saving, we note that both the delay performance evaluation and the path determination in the service chaining engine increase the loading of the SDN controller. Since the topology and the traffic characteristic in service chaining are static, the delay performance evaluation can be performed offline leaving only the CSP problem to be solved online, thereby reducing the workload as well as the energy consumption of the SDN controller. Furthermore, the service chaining engine can be designed as a virtual network function running on a separate VM. On the other hand, resource overbooking is another way to save energy in data-centers, which can reduce the usage of hosts and networks by serving more requests with the same amount of resources [44]. We highlight that the proposed delay performance evaluation can help decide overbooking ratio with fewer delay requirement violations.

IX. CONCLUSION

We investigated a deterministic QoS guarantee issue of service chaining in SDN. We presented a systematic methodology for evaluating the delay performance of an SDN controller. We analyzed the average delay performance of service chaining decision based on queuing theory and derived a deterministic delay bound of service chaining decision according to network calculus (NC). Subject to the constraint of service chain traversal delay derived from the delay bound, an optimal service chaining path determination problem was formulated and solved. Through experiments and analysis, our results demonstrated that the proposed delay upper bound in service chaining is accurate for different controller loads, network scales, and traffic generation rates. As a result, QoS-guaranteed service chaining can be achieved by systematically evaluating the delay performance and designing the service traversal path. In the future, we will extend the proposed NC approach to the dynamic end-to-end network slicing 5G wireless system.

REFERENCES

[1] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "Exploiting congestion games to achieve distributed service chaining in NFV networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 2, pp. 407–420, Feb. 2017.

[2] P.-C. Lin, Y.-D. Lin, C.-Y. Wu, Y.-C. Lai, and Y.-C. Kao, "Balanced service chaining in software-defined networks with network function virtualization," *IEEE Comput.*, vol. 49, no. 11, pp. 68–76, Nov. 2016.

[3] M. Mechtri, C. Ghribi, and D. Zeghlache, "A scalable algorithm for the placement of service function chains," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 533–546, Sep. 2016.

[4] Y.-D. Lin, P.-C. Lin, C.-H. Yeh, Y.-C. Wang, and Y.-C. Lai, "An extended SDN architecture for network function virtualization with a case study on intrusion prevention," *IEEE Netw.*, vol. 29, no. 3, pp. 48–53, May/Jun. 2015.

[5] R.-H. Gau and P.-K. Tsai, "SDN-based optimal traffic engineering for cellular networks with service chaining," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Doha, Qatar, 2016, pp. 1–6.

[6] *Cisco Data Center Infrastructure 2.5 Design Guide*. Accessed: Dec. 10, 2013. [Online]. Available: http://www.cisco.com/univercd/cc/td/doc/solution/dcidg21.pdf

[7] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2317–2346, 4th Quart., 2015.

[8] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

[9] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, Feb. 2013.

[10] X.-N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "Rules placement problem in OpenFlow networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1273–1286, 2nd Quart., 2016.

[11] K.-T. Kuo, C. H.-P. Wen, C. Suo, and I.-C. Tsai, "SWF: Segmented wildcard forwarding for flow migration in OpenFlow datacenter networks," in *Proc. IEEE Int. Conf. Commun.*, London, U.K., 2015, pp. 313–318.

[12] K. Sood, S. Yu, and Y. Xiang, "Performance analysis of software-defined network switch using $M/Geo/1$ model," *IEEE Commun. Lett.*, vol. 20, no. 12, pp. 2522–2525, Dec. 2016.

[13] H. Xu and B. Li, "RepFlow: Minimizing flow completion times with replicated flows in data centers," in *Proc. IEEE Conf. Comput. Commun.*, Toronto, ON, Canada, 2014, pp. 1581–1589.

[14] W. E. Leland, W. Willinger, M. S. Taqqu, and D. V. Wilson, "On the self-similar nature of Ethernet traffic," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 23, no. 4, pp. 183–193, 1993.

[15] Q. Duan, "Network-as-a-service in software-defined networks for end-to-end QoS provisioning," in *Proc. IEEE Wireless Opt. Commun. Conf.*, Newark, NJ, USA, 2014, pp. 1–5.

[16] N. Petreska, H. Al-Zubaidy, R. Knorr, and J. Gross, "On the recursive nature of end-to-end delay bound for heterogeneous wireless networks," in *Proc. IEEE Int. Conf. Commun.*, London, U.K., 2015, pp. 5998–6004.

[17] D. A. Joseph, A. Tavakoli, and I. Stoica, "A policy-aware switching layer for data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 51–62, 2008.

[18] H.-Y. Lam, S. Zhao, K. Xi, and H. J. Chao, "Hybrid security architecture for data center networks," in *Proc. IEEE Int. Conf. Commun.*, Ottawa, ON, Canada, 2012, pp. 2939–2944.

[19] A. Gember, T. Benson, and A. Akella, "Challenges in unifying control of middlebox traversals and functionality," in *Proc. Large Scale Distrib. Syst. Middleware*, 2012, pp. 1–2.

[20] L. Guo, J. Pang, and A. Walid, "Dynamic service function chaining in SDN-enabled networks with middleboxes," in *Proc. IEEE Int. Conf. Netw. Protocols*, Singapore, 2016, pp. 1–10.

[21] A. Gushchin, A. Walid, and A. Tang, "Enabling service function chaining through routing optimization in software defined networks," in *Proc. IEEE Annu. Allerton Conf. Commun. Control Comput.*, Monticello, IL, USA, 2015, pp. 573–581.

[22] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc. USENIX Workshop Hot Topics Manag. Internet Cloud Enterprise Netw. Services*, San Jose, CA, USA, 2012, p. 10.

[23] K. Phemius and M. B. Thales, "OpenFlow: Why latency does matter," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag.*, Ghent, Belgium, 2013, pp. 680–683.

[24] A. Blenk, A. Basta, J. Zerwas, M. Reisslein, and W. Kellerer, "Control plane latency with SDN network hypervisors: The cost of virtualization," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 366–380, Sep. 2016.

[25] S. Azodolmolky *et al.*, "An analytical model for software defined networking: A network calculus-based approach," in *Proc. IEEE Glob. Commun. Conf.*, Atlanta, GA, USA, 2013, pp. 1397–1402.

[26] Y.-J. Chen, F.-Y. Lin, L.-C. Wang, and B.-S. Lin, "A dynamic security traversal mechanism for providing deterministic delay guarantee in SDN," in *Proc. IEEE SDN Archit. Appl.*, Sydney, NSW, Australia, 2014, pp. 1–6.

[27] J. D. C. Little, "A proof for the queuing formula: $L = \lambda w$," *Oper. Res.*, vol. 9, no. 3, pp. 383–387, 1961.

[28] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*. Hoboken, NJ, USA: Wiley, 2008.

[29] M. Ghobadi, S. H. Yeganeh, and Y. Ganjali, "Rethinking end-to-end congestion control in software-defined networks," in *Proc. 11th ACM Workshop Hot Topics Netw.*, Redmond, WA, USA, 2012, pp. 61–66.

[30] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 7, pp. 1228–1234, Sep. 1996.

[31] Y. Xiao, K. Thulasiraman, G. Xue, and M. Yadav, "QoS routing under multiple additive constraints: A generalization of the LARAC algorithm," *IEEE Trans. Emerg. Topics Comput.*, vol. 4, no. 2, pp. 242–251, Apr./Jun. 2016.

[32] A. Juttner, B. Szviatovski, I. Mécs, and Z. Rajkó, "Lagrange relaxation based method for the QoS routing problem," in *Proc. IEEE INFOCOM*, Anchorage, AK, USA, 2001, pp. 859–868.

[33] *Floodlight*. Accessed: May 5, 2013. [Online]. Available: http://floodlight.openflowhub.org

[34] *OpenWrt Linux Distribution for Embedded Devices*. Accessed: Jul. 6, 2013. [Online]. Available: https://openwrt.org

[35] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[36] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw.*, Monterey, CA, USA, 2010, Art. no. 19.

[37] Libvirt Team. *Libvirt: The Virtualization API*. Accessed: Nov. 8, 2013. [Online]. Available: http://libvirt.org

[38] A. Azarfar, J.-F. Frigon, and B. Sansò, "Delay analysis of multichannel opportunistic spectrum access MAC protocols," *IEEE Trans. Mobile Comput.*, vol. 15, no. 1, pp. 92–106, Jan. 2016.

[39] C. A. Garcia-Perez and P. Merino, "Enabling low latency services on LTE networks," in *Proc. IEEE Int. Workshops Found. Appl. Self Syst.*, Augsburg, Germany, 2016, pp. 248–255.

[40] A. S. Muqaddas, A. Bianco, P. Giaccone, and G. Maier, "Inter-controller traffic in ONOS clusters for SDN networks," in *Proc. IEEE Int. Conf. Commun.*, Kuala Lumpur, Malaysia, 2016, pp. 1–6.

[41] B. P. R. Killi and S. V. Rao, "Capacitated next controller placement in software defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 3, pp. 514–527, Sep. 2017.

[42] T. Mizrahi and Y. Moses, "Time4: Time for SDN," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 433–446, Sep. 2016.

[43] T. Mizrahi, O. Rottenstreich, and Y. Moses, "TimeFlip: Using timestamp-based TCAM ranges to accurately schedule network updates," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 849–863, Apr. 2017.

[44] J. Son, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "SLA-aware and energy-efficient dynamic overbooking in SDN-based cloud data centers," *IEEE Trans. Sustain. Comput.*, vol. 2, no. 2, pp. 76–89, Apr./Jun. 2017.

Yu-Jia Chen received the B.S. and Ph.D. degrees in electrical engineering from National Chiao Tung University, Taiwan, in 2010 and 2016, respectively, where he is currently a Post-Doctoral Fellow. He has published 16 conference papers and 3 journal papers. He is holding two U.S. patent and two ROC patent. His research interests include network coding for secure storage in cloud datacenters, software defined networks, and sensors-assisted applications for mobile cloud computing.

Li-Chun Wang (M'96–SM'06–F'11) received the B.S. degree from National Chiao Tung University, Taiwan, in 1986, the M.S. degree from National Taiwan University in 1988, and the Ms.Sci. and Ph.D. degrees from the Georgia Institute of Technology, Atlanta, in 1995, and 1996, respectively, all in electrical engineering.

From 1990 to 1992, he was with the Telecommunications Laboratories, Chunghwa Telecom Company. In 1995, he was affiliated with Bell Northern Research of Northern Telecom, Inc., Richardson, TX, USA. From 1996 to 2000, he was with AT&T Laboratories, where he was a Senior Technical Staff Member with the Wireless Communications Research Department. Since 2000, he has been with the Department of Electrical and Computer Engineering, National Chiao Tung University, where he is currently the Chairman. His current research interests are in the areas of radio resource management and cross-layer optimization techniques for wireless systems, heterogeneous wireless network design, and cloud computing for mobile applications.

Dr. Wang has published over 200 journal and international conference papers. He holds 10 U.S. patents. He was a recipient of the Distinguished Research Award of National Science Council, Taiwan, in 2012, and was elected to the IEEE Fellow Grade in 2011 for his contributions to cellular architectures and radio resource management in wireless networks. He was a co-recipient (with G. L. Stuber and C.-T. Lea) of the 1997 IEEE Jack Neubauer Best Paper Award for his paper "Architecture Design, Frequency Planning, and Performance Analysis for a Microcell/Macrocell Overlaying System," in IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He served as an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS from 2001 to 2005 and the Guest Editor of Special Issue on "Mobile Computing and Networking" for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS in 2005, "Radio Resource Management and Protocol Engineering in Future Broadband Networks" for the *IEEE Wireless Communications Magazine* in 2006, and "Networking Challenges in Cloud Computing Systems and Applications," for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS in 2013.

Feng-Yi Lin was born in Taiwan, in 1989. He received the B.S. degree in computer science from National Chiayi University in 2011 and the M.S. degree from National Chiao Tung University in 2014. His current research interests include software defined networking and network security.

Bao-Shuh Paul Lin received the Ph.D. degree in computer science from the University of Illinois at Chicago, IL, USA. He has been a Chair Professor with the Department of Computer Science and the Chief Director of Microelectronics and Information Systems Research Center, National Chiao Tung University, Hsinchu, Taiwan, since 2009. He has also been the Director of Committee of Communication Industry Development, Ministry of Economic Affairs since 2001. He was the Vice President of Industrial Technology Research Institute (ITRI) from 2001 to 2009 and the General Director of the Information and Communications Research Laboratories (ICL), ITRI from 2001 to 2009. From 1979 to 1991, he was with Bell Labs of Lucent Technologies, Boeing, and two other high-tech firms before coming back to Taiwan in 1991. From 1991 to 1998, he worked for ITRI and served as the Director of Computer Communications Research Division in CCL (the forerunner of ICL) and was later promoted to its Deputy General Director.