SACO: A Service Chain Aware SDN Controller-Switch Mapping Framework

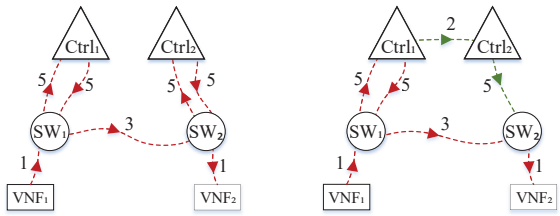
Duong Tuan Nguyen, Chuan Pham, Kim Khoa Nguyen, Mohamed Cheriet
Ecole de technologie Superieure, University of Quebec
Montreal, Quebec, Canada, H3C 1K3
{duong-tuan.nguyen.1,chuan.pham.1}@ens.etsmtl.ca,{kim-khoa.nguyen,mohamed.cheriet}@etsmtl.ca

***Abstract*—The emerging paradigm of Software Defined Network (SDN) and virtualization technology promises an efficient solution for network providers to deploy services. Adopting them not only facilitates network management but also helps reduce the cost of maintaining network infrastructure. However, despite these advantages, there are still obstacles that must be overcome before SDN and virtualization can advance to reality in industrial deployments. In this paper, we focus on two well-researched issues, namely controller-switch assignment and Virtual Network Function (VNF) placement. Unlike prior works, our purpose is to jointly solve these two problems, accounting for the complex and counter-intuitive manner they are related to each other. We present a service chain aware framework (SACO) that enables the controller-switch association in a multi-controller network regarding the relationship of switches via their connected VNFs that implement service components of the chain. We also propose a model and formulate the joint optimization problem of dynamic controller-switch mapping and VNF allocation. We apply the Lyapunov optimization framework to transform a long-term optimization problem into a series of real-time problem and employ the Markov approximation method to find a near-optimal solution. Simulation results show that our service chain aware approach improves the system cost up to 10 $\sim$ 43% compared to the state-of-the-art solutions.**

I. Introduction

Software-defined Networking (SDN) paradigm has drawn great attention lately from both academic communities and industries thanks to the shift of network control from distributed protocols to a logically centralized control plane. From the network perspective, the physical separation of control plane and forwarding plane provides network operators with efficient use of network resources and eases the resource provisioning. From the service point of view, the combination of SDN and Network Function Virtualization (NFV) facilitates service chain provisioning. The rules defined by SDN controllers manage and route traffic of service components that are typically implemented as VNF through switches. In other words, the entire service chain can be provisioned or configured from the controller. Moreover, the NFV's ability to build network functions as software means that deploying a chain of service components is more dynamic due to no longer requires hardware.

In an effort to mitigate the resilient issue of a single point of failure and to improve scalability, extensive research work is being carried out in proposing distributed architecture with multiple controllers [1]. However, this results in another question of how to effectively map switches to controllers taking into account workload at network elements as well as Quality of Service (QoS) requirements. A few recent attempts have explored the limits of the static assignment between a switch and a controller, i.e. difficult for the control plane to adapt to traffic load fluctuation, by tackling the dynamic controller/switch mapping problem. Typical approaches include designing a migration protocol to migrate a switch among controllers [1], formulating as an optimization problem aiming to optimize controller response time, traffic overhead [2], [3], [4]. Although the enhancement of controller performance has been shown via experimental results, we argue that only regarding data or control plane might make the improvement appear to be as a case of "winning the battles but losing the war". To illustrate our point of how the awareness of service chain is important, let us consider flow setup latency of the example in Fig. 1. With the approach aligning with OpenFlow specification [5] in Fig. 1a, to setup a session between VNF_1 and VNF_2, the traffic must go through each controller from the corresponding switch and take 25 units of time to complete. However, with the prior knowledge of the service chain, upon receiving the first packet from switch 1, controller 1 can inform controller 2 to install an appropriate forwarding rule at switch 2. By doing so, switch 2 does not need query controller 2 when the traffic arrives. In Fig. 1b, the total delay is only 10 units of time.



(a) Normal flow setup (b) Service aware flow setup

Fig. 1. Impact of SDN controllers' operations to flow setup latency

In this paper, we consider a data-center network where service chains are deployed in the form of VNFs as service components. VNFs are allocated on blade servers hosted by racks. OpenFlow-enabled switches are assumed to reside in either the server or the Top-of-Rack switch and handle traffic from VNFs. Each switch may connect to more than one SDN

978-3-903176-24-9

controller. Our goal is to find an efficient method to i) assign connections between controllers and switches and ii) place VNFs under the control of switches regarding network resource, service latency, flow setup delay, and system stability.

The paper contribution is three-fold. First, we design a Service Chain Aware SDN Control-Switch Mapping framework (SACO) that allows multiple controllers to communicate with each other about the presence of service chain in order to improve the QoS. Second, we introduce a model for the problem of optimally mapping network elements, i.e. controllers and switches, and placing VNFs onto racks handled by switches. We formulate it as a non-convex Binary Integer Programming problem. Third, to address highly fluctuated service demand which is typically unpredictable, we apply Lyapunov optimization framework [6] to transform this long-term optimization problem into a series of real-time problem without a *priori* knowledge. Regarding the NP-hardness of transformed combinatorial network optimization problem, we employ Markov approximation method [7] to find a near-optimal solution. The evaluation via simulations verifies the effectiveness of the proposed algorithm over those that do not take into account the presence of service chains in terms of switches' relationship.

The remainder of the paper is organized as follows. After reviewing related work in Section II, we introduce the SACO framework and coordination protocol in Section III. The model and problem formulation are presented in Section IV while Section V explains the proposed algorithms. Section VI provides a performance evaluation of our methods with simulation settings. Finally, conclusions are drawn.

II. Related Work

In multi-SDN-controller networks, the problem of dynamically assigning the connections between switches and controllers has been well investigated in the literature. Many solutions have been proposed with consideration on the vital role of the control plane. The authors in [1] propose an elastic distributed controller architecture that allows migration of OpenFlow switches between different controllers. However, they do not detail how to achieve an efficient migration, especially in case one switch may have connections to multiple controllers.

Reference [2] presents a dynamic assignment scheme with a goal to balance the controller load while keeping the control traffic overhead low. Apart from load balance at control plane, reference [8] proposes switch migration schemes considering the migration cost whereas [9] approaches the problem from game theory perspective by formulating it as a one-to-many matching game with a minimum quota of controller's processing capacity. Similarly, also taking switch migration into account, Xing *et al.* in [4] design a scalable control mechanism subject to maximize control resource utilization. The approach studied in [10] attempts to obtain a strategy to jointly optimize controller-switch assignment and control-path routing to minimize the number of recovery stages. To reduce flow setup time, reference [3] introduces a multiple mapping approach that allows a switch to distribute flow setup requests to multiple controllers.

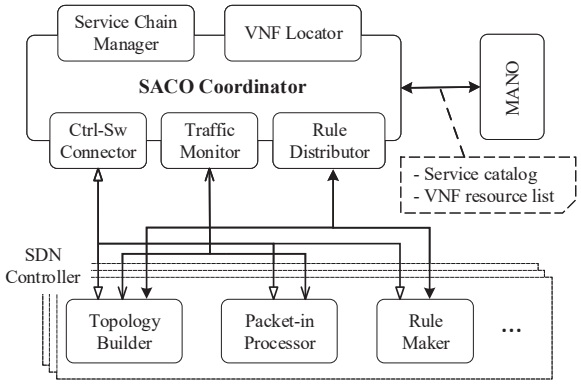


Fig. 2. SACO System Architecture

Closer to our work is [11] which aims to find a dynamic switch-controller association so as to minimize the long-term average cost. However, the authors do not consider the constraints of resource as well as service latency, which have a significant impact on the success of service delivery.

In this work, we tackle the controller-switch assignment with regard to the presence of the service chain. In particular, we exploit the dependence of switches incurred by the traffic going through VNFs as service components, which has not been taken into account in previous studies.

III. Service Chain Aware Controller-Switch Mapping Architecture

In this section, we detail the system with SACO Coordinator and the message flows between the coordinator with multiple controllers with the presence of connected OpenFlow switches.

A. SACO System Architecture

The overall SACO system architecture is shown in Fig. 2. In general, the SACO Coordinator coordinate multiple SDN controllers to make sure that forwarding rules are appropriately installed onto OpenFlow switches with regard to service chains. The idea is that when a service chain' flow arrived, this architecture allows the responsible controller to inform the others to make the rules in advance so as to reduce total communication delay. We assume that any SDN controller at the control plane can be reached from the others via an East-West API interface [12].

The Coordinator stays on top of SDN controllers and interacts with NFV Management & Orchestration (MANO) with corresponding components. For instance, Service Chain Manager communicates with MANO for the information of service chain, in terms of the template, QoS requirements, which can be extracted from network service catalog. The MANO is also able to provide the details of VNF resource so that VNF Locator can determine where to place VNFs.

Towards SDN controller side, there are three main components interacting with the SACO Coordinator, namely Topology

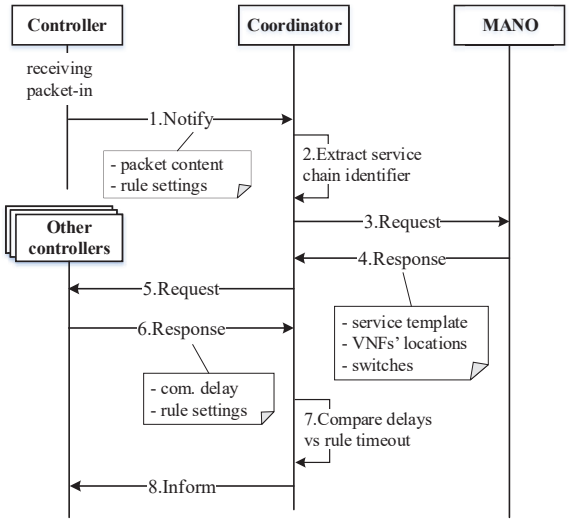


Fig. 3. SACO Coordination Protocol

Builder (TB), Packet-in Processor (PP) and Rule Maker (RM). Note that these functionalities are widely implemented in SDN controllers, i.e. ONOS [13]. The connections between OpenFlow switches and controllers are handled by Ctrl-Sw Connector at the moment determined by the PP, with the topology input given by the TB, and performed via rules by RM. Traffic Monitor collects traffic-related data for the inputs of SCAMP optimization algorithms, i.e. arrival rates, communication delay between network elements. Upon receiving the algorithm outputs, Rule Distributor allows the Coordinator to communicate with multiple controllers to apply relevant rules. Note that in case of failure, an automatic disconnecting mechanism could be used to isolate the Coordinator to avoid the single point of failure.

B. SACO Coordination Protocol

Fig. 3 illustrates how SACO coordinator interacts with SDN controllers and MANO implementation to install traffic policies in advance so as to reduce the total service chain's delay. When a certain controller receives a packet of new flow's traffic, a Notification message with the packet's content and the controller's rule settings is sent to SACO (Step 1). The packet's content provides service chain's information, i.e. service path identifier [14] while rule settings are related to timeout value, i.e. soft/hard timeout, maximum valid duration.

With extracted service path identifier (Step 2), the Coordinator sends a request (Step 3 & 4) to MANO for the details of the service chain corresponding to the packet, in terms of its structure, relevant VNFs' location, and OpenFlow switches that directly handle traffic of the VNFs. Such the information can be collected via interfaces between MANO with SDN controllers or VNF managers and out of the scope of this paper.

The Coordinator also requires the link latency between any two elements, either switches or controllers and the rule settings of the controllers that manage the switches returned

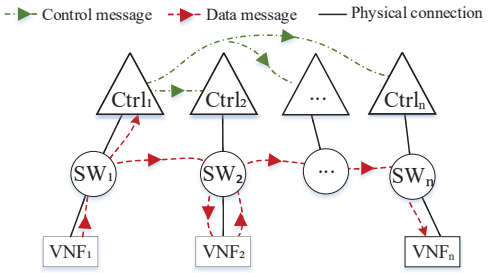


Fig. 4. Illustration of SACO Rule Distribution

by MANO at Step 5. Such the information is the input of the procedure at Step 7 which decides whether or not the Coordinator should inform other controllers to update their rules to involved switches.

To illustrate this idea, we consider a simple scenario in Fig. 4 with a flow of a service chain going from VNF_1 to VNF_n and each VNF connects to an OpenFlow switch handled by only one SDN controller. Upon receiving a packet, switch 1 forwards it to controller 1 which attaches that packet information into a request sent to the Coordinator (not shown in the figure for simply illustrating). Controller 1 only informs controller n to update a rule related to the flow if the summation of rule timeout value configured for switch n and the delay between Ctrl_1 and Ctrl_n is smaller than the delay between switch 1 and switch n. By doing so, it makes sure that when the traffic arrives at switch n, there is always a matching rule and as a result, it does not take so long time for the switch to ask the controller.

C. Discuss

While service chain's latency can be reduced by coordinating SDN controllers to setup flow rules in advance, the SACO framework is based on an existing controller-switch mapping and VNF placement scheme. In details, the response from MANO at Step 4 in Fig. 3 or from SDN controllers at Step 6 must be available.

In the rest of this paper, we provide a solution to retrieve such the scheme which is formulated as an optimization problem. Note that the proposed algorithms to solve this problem can be implemented at the SACO Coordinator and its output can be locally managed by the SACO itself rather than being stored by other components, i.e. SDN controllers, MANO. However, in a large-scale network, the interaction between it with MANO or SDN controller is necessary to avoid placing too much workload at the Coordinator.

IV. System Description & Modeling

In this section, we describe the system and provide a model of SACO problem regarding the proposed architecture and coordination protocol.

A. System Modeling

We consider a network that is composed of a set $G = \{C \cup S\}$ nodes where C is the set of controllers at control plane and S

represents a set of switches at data plane. Each switch $s \in S$ may have several interfaces to connect to more than one controllers at a time. It is also responsible of forwarding traffic from a subset of service functions implemented as VNFs of the set F are used to construct a set H of service chains. The construction of each service chain h is pre-defined via $\alpha^h_{v,v'}$ where $\alpha^h_{v,v'} = 1$ if v and v' are consecutively executed in service chain h, and 0 otherwise. Using the same notation, $\alpha^h_v = \sum_{v' \in V} \alpha^h_{v,v'}$ to determine whether or not v is used by h. The system is assumed to operate in slotted time $t \in \{0, 1, 2, \ldots\}$. The placement of the VNF v onto the rack controlled by s is represented by a binary variable $x^s_v(t) \in \{0, 1\}$. The connection between a switch s and a controller c on slot t is determined by another variable $y^c_s(t) \in \{0, 1\}$. In addition, due to the presence of multiple managing controllers, a switch schedules to forward packets of any new flow from a certain VNF v to one of its connected controllers. The rule timeout decided by the controller for flows of the service chain h is defined at Δ_h. Let $z^c_v(t) = 1$ imply that the controller c processes traffic from v. We have following constraints:

$$\sum_{s \in S} x^s_v(t) \leq 1, \forall v \in F \tag{1}$$

$$\sum_{s \in S} x^s_v(t) y^c_s(t) \leq z^c_v(t), \forall v \in F, c \in C \tag{2}$$

1) Network Resource Model: Regarding network resource capacity, each service chain h has different requirement on its VNFs v, i.e. r^h_v. The total bandwidth allocated for VNFs should not exceed the capacity of the corresponding switch, i.e. r_s, or

$$\sum_{(h,v) \in (H,F)} r^h_v x^s_v(t) \leq r_s, \forall s \in S \tag{3}$$

Similarly, a controller with its maximum available bandwidth resource, r_c, is only able to manage a limited number of switches, which is denoted as

$$\sum_{s \in S} r_s y^c_s(t) \leq r_c, \forall c \in C \tag{4}$$

2) Service Latency Model: The communication delay between any two network nodes is denoted as $d_{e,e'}$ with $e, e' \in G$. To simplify the model, the delay between a VNF with any connected switch is assumed to have negligible impact on the total latency of a service chain and therefore ignored from our model. This assumption is reasonable given the fact that VNFs and the corresponding top-of-rack (ToR) switches are typically placed on same racks. We consider the total delay constraint of service chain at each slot, i.e. $d_h(t)$, that is, $\forall h \in H$

$$d_h(t) = \sum_{v,v' \in V} \sum_{s,s' \in S} \alpha^h_{v,v'} d_{s,s'} x^s_v(t) x^{s'}_{v'}(t) \leq \Phi^D_h. \tag{5}$$

where Φ_h is the maximum tolerable delay of the service chain h.

Apart from the total latency, we also take into account flow setup latency of a service chain which plays an important role in measure the quality of many applications, e.g. SIP-based services. According to the proposed protocol described in Section III-B, this metric denoted as $f_h(t)$ and its constraint are modeled as follows

$$f_h(t) = \sum_{v,v' \in F} \sum_{s,s' \in S} \sum_{c,c' \in C} \alpha^h_{v,v'} x^s_v(t) x^{s'}_{v'}(t) * z^c_v(t) z^{c'}_{v'}(t)(d_{s,c} + d_{c,c'} + d_{s',c'}) \leq \Phi^F_h, \forall h \in H \tag{6}$$

3) System Stability Model: Let $\lambda_h(t)$ denote the service arrival rate of service chain h, $\mu_s(t)$ and $\mu_c(t)$ respectively be the number of packets s and c can process on slot t. It is assumed that VNFs operate as M/M/1 queuing systems. The service arrival rates are obtained as follows:

$$\lambda_v(t) = \sum_{h \in H} \alpha^h_v \lambda_h(t), \forall v \in F \tag{7}$$

$$\lambda_s(t) = \sum_{v \in F} \lambda_v(t) x^s_v(t) + \sum_{c \in C} y^c_s(t), \forall s \in S \tag{8}$$

$$\lambda_c(t) = \sum_{h \in H} \sum_{v \in F} \frac{1}{\Delta_h} \alpha^h_v z^c_v(t), \forall c \in C \tag{9}$$

Let $Q_s(t)$ and $Q_c(t)$ be current queue backlog of switch s and controller c on slot t, respectively and recursively given based on previous time slot by, $\forall e \in G$:

$$Q_e(t+1) = max[Q_e(t) - \mu_e(t), 0] + \lambda_e(t). \tag{10}$$

According to [6], a queueing system is stable if and only if the following condition holds:

$$\lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{e \in G} E\{Q_e(\tau)\} < \infty \tag{11}$$

4) System Cost Model: From the above analysis, the system cost on slot t, i.e. $m(t)$ can be formulated as

$$m(t) = \sum_{s \in S} \left(m^{dta}_s \lambda_s(t) + \sum_{c \in C} m^{ctl}_s y^c_s(t) \lambda_c(t) \right) \tag{12}$$

where m^{dta}_s and m^{ctl}_s are respectively the cost to transmit a unit of traffic, i.e. packet, at the data and control plane.

In this paper, we consider the time average total resource cost for serving the delay-sensitive service requests over a large time horizon, that is

$$\bar{m} = \limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E\{m(t)\} \tag{13}$$

5) Problem Formulation: We are interested in the problem stated as follows: for the dynamic system defined by (10), design a switch-controller assignment and VNF allocation scheme which, given the present information in terms of resource requirements of network elements as well as communication delay between them, resource cost, service arrivals, chooses the placement decisions $\mathbf{x} = \{x^s_v | (s, v) \in (S, F)\}$, the assignment decisions $\mathbf{y} = \{y^c_s | (c, s) \in (C, S)\}$ and the routing decisions of

flow setup message $\mathbf{z} = \{z_v^c | (c, v) \in (C, F)\}$ to minimize $\bar{m}$. It can be formulated as the following stochastic optimization:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y},\mathbf{z}}{\text{minimize}} \quad & \bar{m} && (14) \\ \text{subject to} \quad & (1)-(6), (11) \\ & x_v^s(t), y_s^c(t), z_v^c(t) \in \{0,1\}, \\ & \qquad \forall v \in F, s \in S, c \in C && (15) \end{aligned}$$

V. Algorithm Design and Analysis

In this section, we develop the SACO's algorithm to solve the problem (14). We begin by explaining how Lyapunov optimization technique is adopted to construct another version of the long-term original problem, which can be solved in each time slot. Then relaxation techniques are applied to make the modified problem formulation convex.

A. Lyapunov Optimization Framework

Given a system queue backlog vector $\mathbf{\Theta}(t) = [Q_e(t)]_{e \in G}$, we define the Lyapunov function and one-step conditional Lyapunov drift for slot t as follows:

$$L(\mathbf{\Theta}(t)) = \frac{1}{2} \sum_{e \in G} Q_e(t)^2 \tag{16}$$

$$\Delta(\mathbf{\Theta}(t)) = E\Big\{ L\big(\mathbf{\Theta}(t+1)\big) - L\big(\mathbf{\Theta}(t)\big) | \mathbf{\Theta}(t) \Big\} \tag{17}$$

Intuitively, $\Delta(\mathbf{\Theta}(t))$ is the difference of queue length between two consecutive slot. Therefore the closer a SACO algorithm pushes this value towards its bound, the more stable the system queue is. We have following theorem to determine such the bound.

Theorem 1. *For any queue backlog $\mathbf{\Theta}(t)$ under any placement scheme, $\Delta(\mathbf{\Theta}(t))$ is upper bounded by*

$$\begin{aligned} \Delta(\mathbf{\Theta}(t)) \leq \frac{1}{2} E\Big\{ \sum_{e \in G} \big(\lambda_e(t)^2 + \mu_e(t)^2\big) | \mathbf{\Theta}(t) \Big\} + \\ E\Big\{ \sum_{e \in G} Q_e(t) \lambda_e(t) | \mathbf{\Theta}(t) \Big\} - \sum_{e \in G} Q_e(t) \mu_e(t) \end{aligned} \tag{18}$$

Proof. Using the definition of $L(\mathbf{\Theta}(t))$ from (10) and substituting (16) into (17) yield:

$$\begin{aligned} L(\mathbf{\Theta}(t+1)) - L(\mathbf{\Theta}(t)) &= \frac{1}{2} \sum_{e \in G} \big(Q_e(t+1)^2 - Q_e(t)^2 \big) \\ &= \frac{1}{2} \sum_{e \in G} \Big[(max[Q_e(t) - \mu_e(t), 0] + \lambda_e(t))^2 - Q_e(t)^2 \Big] \\ &\leq \sum_{e \in G} \Big[\frac{\lambda_e(t)^2 + \mu_e(t)^2}{2} + Q_e(t) \big(\lambda_e(t) - \mu_e(t) \big) \Big] \end{aligned} \tag{19}$$

Taking the expectation of two sides in (19) proves (18), that is

$$\begin{aligned} \Delta(\mathbf{\Theta}(t)) &\leq B + E\Big\{ Q_e(t) \big(\lambda_e(t) - \mu_e(t) \big) \Big] | \mathbf{\Theta}(t) \Big\} \\ &= B + E\Big\{ \sum_{e \in G} Q_e(t) \lambda_e(t) | \mathbf{\Theta}(t) \Big\} - \sum_{e \in G} Q_e(t) \mu_e(t) \end{aligned} \tag{20}$$

where $B = E\big\{ \sum_{e \in G} \big[\frac{\lambda_e(t)^2 + \mu_e(t)^2}{2} | \mathbf{\Theta}(t) \big\}$ and the fact that the service processing rate $\mu_e(t), \forall e \in G$ is independent of current queue backlog. □

However, while minimizing a bound on $\Delta(\mathbf{\Theta}(t))$ would stabilize the system and satisfy constraint (11), it may result in a high cost $\bar{m}$. Instead, by incorporating queue stability into system cost, we introduce a Lyapunov drift-plus-penalty function as follows

$$\Delta(\mathbf{\Theta}(t)) + VE\{m(t) | \mathbf{\Theta}(t)\} \tag{21}$$

where V is a control parameter to determine how much cost minimization is emphasized. Then from Theorem 1, the objective function of problem (14) can be re-formulated in combining with constraint (11). The problem is re-stated as: every slot t, given the current queue states $\mathbf{\Theta}(t)$, $\mathbf{x}, \mathbf{y}, \mathbf{z}$ must satisfy constraints (1)-(6) and

$$\underset{\mathbf{x},\mathbf{y},\mathbf{z}}{\text{minimize}} \quad Vm(t) + \sum_{e \in G} Q_e(t) (\lambda_e(t) - \mu_e(t)) \tag{22}$$

The implementation of online SACO algorithm is summarized in Alg. 1. Note that the SACO algorithm does not require any prior knowledge of the service arrival process as well as the placement decision in the past, which makes it applicable for unpredictable service demands. Unfortunately, the problem (22) is still NP-hard. It is difficult to find the optimal solution in the polynomial time, especially in practical settings with a high number of VNFs, network switches and controllers. To tackle this issue, we adopt a Markov-based approximation method proposed in [7].

B. Markov-based Approximation Framework

1) Log-Exp-Sum Approximation: Let $f = \{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ be a specific VNFs placing and controllers-switches assignment scheme. We denote $\mathcal{F}$ as the set of feasible schemes (or configurations) defined by constraints (1)-(6). Given an existing scheme f, there are several operations to generate a new scheme f': i) migrating a VNF from one switch to another, ii) adding or removing a connection between a switch and a controller, and iii) assigning the traffic of a certain VNF to another controller. The system cost under a configuration f is named as m_f with the time slot omitted in the notation for brevity. The problem (22) is re-written as follows:

$$\underset{\mathbf{p} \geq 0}{\text{minimize}} \quad \sum_{f \in \mathcal{F}} p_f m_f \tag{23}$$

$$\text{s.t.} \quad \sum_{f \in \mathcal{F}} p_f = 1 \tag{24}$$

where p_f is the probability of choosing scheme f. The problem (23) is approximated given a positive constant δ as pointed out in [7], thus

$$\underset{\mathbf{p} \geq 0}{\text{minimize}} \quad \sum_{f \in \mathcal{F}} p_f m_f + \frac{1}{\delta} \sum_{f \in \mathcal{F}} p_f log(p_f) \tag{25}$$

$$\text{subject to} \sum_{f \in \mathcal{F}} p_f = 1 \tag{26}$$

Algorithm 1 Online SACO algorithm

1: Initially set $Q_e(0) \leftarrow 0, \forall e \in G$
2: At each slot t, collect controllers' and switches' queue backlog $Q_e(t)$, processing rate $\mu_e(t)$ and service request $\lambda_v(t)$
3: Solve the problem (22) to obtain $\mathbf{x}, \mathbf{y}, \mathbf{z}$
4: Update the queue backlog according to (10)
5: Set $t \leftarrow t + 1$

Algorithm 2 Markov-based Approximation algorithm

1: Initialize a configure $f = f_0$, i.e. placing VNFs in a rack handled by a switch, connecting a switch to a controller, and assigning controllers to VNFs' process traffic.
2: From f, generate a new placement and assigning scheme f' with a corresponding cost $m_{f'}$.
3: Compute the transition probability based on Eq. (29) and set the best configuration to either the current one f or the newly generated one f'.
4: Go to step 2 until stopping criteria is met.

By solving the Karush-Kuhn-Tucker (KKT) conditions of the problem (25), the optimal probability is retrieved as

$$p^*(\mathbf{m}_f) = \frac{exp(-\delta m_f)}{\sum_{f' \in \mathcal{F}} exp(-\delta m_{f'})}, \forall f \in \mathcal{F} \tag{27}$$

Intuitively, the more optimal a configuration is chosen, the more optimal the system cost is. However, in order to compute p_f^* for each configuration, it requires to take into account the whole feasible configuration space to compute (27), i.e. the sum at the denominator, which is inefficient due to the large solution space $\mathcal{F}$. Instead, a Markov chain is constructed in a way that the stationary distribution of each state is p_f^*. The existence of such the chain has been already proven in [7].

C. Markov Chain Construction Procedure

Let two configurations f, f' in $\mathcal{F}$ represent two states of the time-reversible ergodic Markov chain with the stationary probability $p^*(\mathbf{m}_f)$. The transition probability between f and f', which are $t_{(f \to f')}$ and symmetrically defined $t_{(f' \to f)}$, must satisfy:

$$p^*(\mathbf{m}_f) t_{(f \to f')} = p^*(\mathbf{m}_{f'}) t_{(f' \to f)} \tag{28}$$

There are many values of $t_{(f \to f')}$ and $t_{(f' \to f)}$ in Equ. (28). We choose the following option with $t_{(f \to f')}$ defined symmetrically, which is:

$$t_{(f \to f')} = \rho \; exp\Big(\frac{1}{2}\delta(m_f - m_{f'})\Big) \tag{29}$$

where ρ is a conditional non-negative constant. A basic procedure to construct a Markov chain toward the stationary distribution is thus given in Alg. (2)

VI. Numerical Results

A. Simulation Setup

We illustrate the efficiency of our proposed approach under a data center network with $|C| = 16$ controllers, $|S| = 128$ switches, each of which is responsible for a rack that can host 20 VNFs. The simulation is performed with the number of service chains ranging from 100 to 1000. Each service chain contains from 10 to 100 VNFs. Service arrival process follows the distribution of flow inter-arrival time with mean varied between 1ms and 10ms. Service processing rates at switches and controllers are to $0.1ms^{-1}$ and $0.3ms^{-1}$ respectively. Other simulation inputs are detailed in Table I.

We adopt numerical calculation approach to validate the improvement of proposed approach. From the implementation perspective, each Markov state is composed of matrices that determine i) the placement of VNFs onto racks controlled by certain switches, ii) the assignment of switches to controllers, and iii) the controller that processes traffic from a certain VNF. Network connections between entities are represented by adjacency matrices whose elements are tuples of link delay, link cost and bandwidth. This information is considered as global knowledge shared among SDN controllers.

TABLE I
Simulation parameters

Parameter	Value
Nbr of service chains	{100, 1000}
Service chain's length	{10, 100}
Traffic cost	m_s^{dta} (5, 0.2), m_s^{ctl} (3, 0.2)
Processing rate (ms^{-1})	μ_s (0.1, 0.3), μ_c (0.1,0.3)
Service arrival rate (ms^{-1})	λ_h (0.1, 0.9)
Network bandwidth (Mbps)	r_s^h (1.5,2.5) r_s (0.3,0.7) r_c (0.3,0.7)
Communication delay (ms)	$d_{s,c}$ (1.4,0.02) $d_{s,s'}$ (1,0.02) $d_{c,c'}$ (1,0.02)

B. Comparison with Other Approaches

In order to assess the performances of our algorithm in terms of minimization system cost while satisfying the requirement of total delay as well as flow setup latency, we compare it with three other schemes: Greedy and SCU (service chain unaware). In greedy scheme, at each slot, each VNF is prioritized to connect to the switch with minimum transmission cost at data plane rather than at control plane. A switch is assigned to a certain number controllers with lowest transmission cost. The number of controllers is picked to make sure that resource constraints are satisfied. In SCU scheme, the constraints on service latency and flow setup delay, i.e. (5) & (6) are ignored. In both greedy and SCU schemes, the total cost is calculated based on Eq. (12) and the penalty incurred when a service is rejected due to its high latency. This result can be verified by the service rejection rate at Fig. 6 when the SACO can guarantee a low rejection rate as the service demand increases.

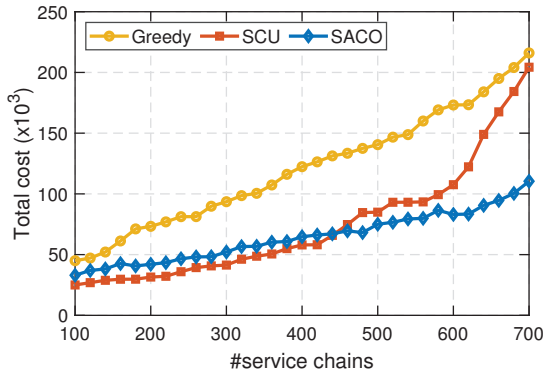


Fig. 5. Total system cost with different service demand

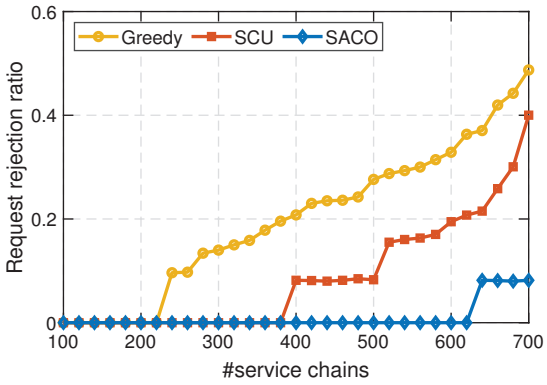
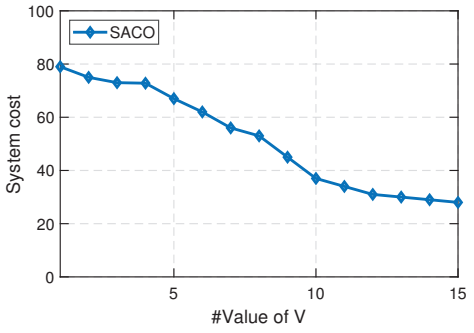


Fig. 6. Service rejection rate with different service demand

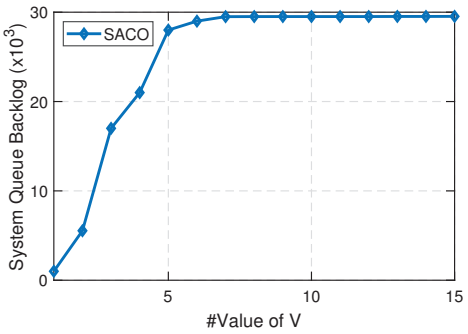
Fig. 5 shows an evaluation of the three schemes in terms of total system cost with different service demand, or the number of service chains. The greedy approach returns a solution whose cost is the summation of a constant m and an increasing penalty cost when more service chains are rejected due to violating QoS requirements. The SCU maintains a stable cost when the number of service chains is less than 600. With more than 600 chains, the penalty cost incurred by service rejection is significantly contributed to the total cost. Meanwhile, the SACO is more costly than the SCU with a low number of service chains. Under a high service demand, the consideration of service latency and flow setup delay helps reduce the service rejection rate and therefore prevents the total cost from rapidly going up. The SACO can save the cost about 11 $\sim$ 43% comparing to the SCU given 700 service chains.

C. Evaluation of Proposed Algorithm

Fig. 7 shows how total system cost and system queue backlog changes with different value of V. As V increases, the cost decreases while the queue goes up in size. In other words, the algorithm places more value on reducing system cost rather than stabilizing system queue. Since V controls the "density" of connections at either switches or controllers, large V means that VNFs are preferably placed together as many as possible in a rack controlled by the same switches and their corresponding



(a) Total system cost



(b) System queue backlog

Fig. 7. Impact of parameter V

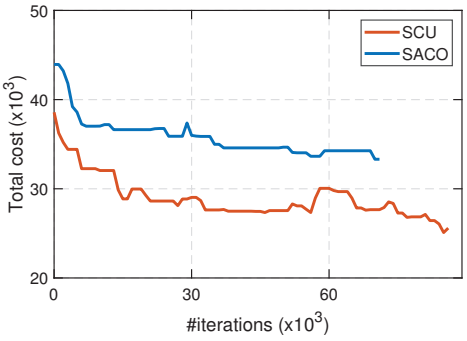


Fig. 8. Algorithm convergence of different approaches

traffic is processed by as few number of controllers as possible. This is consistent with our previous theoretic analysis.

Aiming to tackle the problem in an online manner, the convergence of proposed algorithm plays an important role and is illustrated in Fig. 8. With the presence of additionally constraints and parameters, the SACO is a bit slower than the SCU which does not take into account the service chain aspects.

VII. Conclusion

In this paper, we investigate the problem of allocating the connection between SDN controllers and switches. We propose a SACO framework that enables such the allocation while taking into account the relationship of forwarding network elements, via the presence of service chains. We define an analytical model of system cost in regard to system resources,

service latency, flow setup delay and system stability and formulate our objective that aims to minimize system cost given the input traffic from service components which are typically implemented as VNFs handled by switches and SDN controllers. We present a Lyapunov and Markov approximation based algorithm to obtain the near-optimal solution. The simulated results show that our method can improve the system cost up to $11 \sim 43\%$ compared to the state-of-the-art approaches.

In future, we will validate our work with real world settings. Real data centers with practical service requests will be examined to verify how much the proposed approach can improve the QoS.

ACKNOWLEDGMENT

The authors thank NSERC and Ericsson for funding the project CRDPJ 469977. This research also receives support from the Canada Research Chair, Tier 1, hold by Mohamed Cheriet.

REFERENCES

[1] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an elastic distributed sdn controller," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 7–12, Aug. 2013.

[2] T. Wang, F. Liu, and H. Xu, "An efficient online algorithm for dynamic sdn controller assignment in data center networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2788–2801, Oct 2017.

[3] V. Sridharan, M. Gurusamy, and T. Truong-Huu, "On multiple controller mapping in software defined networks with resilience constraints," *IEEE Communications Letters*, vol. 21, no. 8, pp. 1763–1766, Aug 2017.

[4] X. Ye, G. Cheng, and X. Luo, "Maximizing sdn control resource utilization via switch migration," *Computer Networks*, vol. 126, pp. 69 – 80, 2017.

[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] M. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan Claypool, 2010. [Online]. Available: https://ieeexplore.ieee.org/document/6813406

[7] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Trans. Inf. Theory*, vol. 59, no. 10, pp. 6301–6327, Oct 2013.

[8] Y. Xu, M. Cello, I. Wang, A. Walid, G. Wilfong, C. H. . Wen, M. Marchese, and H. J. Chao, "Dynamic switch migration in distributed software-defined networks to achieve controller load balance," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 515–529, March 2019.

[9] A. Filali, A. Kobbane, M. Elmachkour, and S. Cherkaoui, "Sdn controller assignment and load balancing with minimum quota of processing capacity," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[10] S. S. Savas, M. Tornatore, F. Dikbiyik, A. Yayimli, C. U. Martel, and B. Mukherjee, "Rascar: Recovery-aware switch-controller assignment and routing in sdn," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1222–1234, Dec 2018.

[11] X. Huang, S. Bian, Z. Shao, and H. Xu, "Dynamic switch-controller association and control devolution for sdn systems," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[12] P. Lin, J. Bi, S. Wolff, Y. Wang, A. Xu, Z. Chen, H. Hu, and Y. Lin, "A west-east bridge based sdn inter-domain testbed," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 190–197, Feb 2015.

[13] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "Onos: Towards an open, distributed sdn os," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. New York, NY, USA: ACM, 2014, pp. 1–6.

[14] P. Quinn, U. Elzur, and C. Pignataro, "Network Service Header (NSH)," RFC 8300, Jan. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8300.txt