Joint Switch-Controller Association and Control Devolution for SDN Systems: An Integration of Online Control and Online Learning

Xi Huang[1], Yinxu Tang[1], Ziyu Shao[1*], Yang Yang[1,2,3], Hong Xu[4]
[1]School of Information Science and Technology, ShanghaiTech University
[2] Shanghai Institute of Fog Computing Technology (SHIFT), SCA
[3] Research Center for Network Communication, Peng Cheng Laboratory
[4] Department of Computer Science, City University of Hong Kong
Email: {huangxi, tangyx, shaozy, yangyang}@shanghaitech.edu.cn, henry.xu@cityu.edu.hk

Abstract*—In software-defined networking (SDN) systems, it is a common practice to adopt a multi-controller design and control devolution techniques to improve the performance of the control plane. However, in such systems the decision making for joint switch-controller association and control devolution often involves various uncertainties, *e.g.*, the temporal variations of controller accessibility, and computation and communication costs of switches. In practice, statistics of such uncertainties are unattainable and need to be learned in an online fashion, calling for an integrated design of learning and control. In this paper, we formulate a stochastic network optimization problem that aims to minimize time-average system costs and ensure queue stability. By transforming the problem into a combinatorial multi-armed bandit problem with long-term stability constraints, we adopt bandit learning methods and optimal control techniques to handle the exploration-exploitation tradeoff and long-term stability constraints, respectively. Through an integrated design of online learning and online control, we propose an effective Learning-Aided Switch-Controller Association and Control Devolution (*LASAC*) scheme. Our theoretical analysis and simulation results show that LASAC achieves a tunable tradeoff between queue stability and system cost reduction with a sublinear regret bound over a finite time horizon.

I. Introduction

Over the past decade, software-defined networking (SDN) has emerged to facilitate more efficient network management and more flexible network programmability [1]. The key idea of SDN is to separate the control plane from the data plane. In this way, the control plane maintains a logically centralized view to orchestrate the whole network, leaving the data plane to carry out basic network functions such as monitoring and packet forwarding. In the data plane, the forwarding devices such as SDN-enabled switches [2] constantly generate requests to be processed by the control plane, *e.g.*, to make routing decisions for dedicated flows.

As data plane expands, the control plane may become a bottleneck in SDN systems for scalability and reliability concerns. Existing works adopt two approaches to control plane design. One approach is to implement the control plane with physically distributed controllers. In such a design, each switch may have potential connections to multiple controllers. Due to temporal variations of system dynamics, each switch's accessible controller set may change over time. Accordingly, the key challenge in such design lies in how switches should forward requests amongst their associated controllers, that is *switch-controller association*. To this end, some recent works have proposed various solutions [3]–[12]. The other approach is to delegate part of request processing that requires only local information onto SDN-enabled switches or controllers near the data plane [13]–[16], that is *control devolution*, to mitigate control plane's workload. The key challenge in such design lies in *when* and *whether* switches should, if possible, process requests locally or upload them to the control plane.

The above two approaches are orthogonal to each other and can be further combined. Nonetheless, such a joint design often involves two concerns. One concern is about the *tradeoff* among the communication costs (*e.g.*, bandwidth or round-trip time) incurred by uploading requests to the control plane, local computational costs on switches, and queue stability in SDN systems [17]. The other concern comes from various *uncertainties* in SDN systems. For example, statistics of request traffic and controller accessibility are often not attainable in practice; besides, switches' communication costs incurred or computation costs would be revealed only when the corresponding transmission or processing is completed. Faced with such uncertainties, we need an *integrated* design of online learning and online control. However, there are several challenges. *First*, the learning procedure must deal with the *exploration-exploitation* tradeoff. This is because i) over-exploration, *i.e.*, switches actively forward requests to different controllers, may result in not only balanced and stable queue backlogs but also excessive communication cost; ii) over-exploitation, *i.e.*, switches stick to sending requests to some controllers with empirically lowest costs, may overload such controllers and miss potentially better candidates. *Second*, with online learning and online control procedures being coupled, their interplay demands a careful design, because 1) the learning procedure, if conducted ineffectively, may misguide the control procedure and lead to excessive costs or overloaded queue backlogs; 2) meanwhile, the control procedure, if carried out improperly, may incur noisy feedback

This research was partially supported by the Nature Science Foundation of Shanghai under Grant 19ZR1433900, National Key Research and Development Program of China under Grant YFB0102104, and Huawei Technologies under Grant YBN2019125163. (*Corresponding author: Ziyu Shao)

and thus impose adverse effects on learning efficiency. *Third*, due to various uncertainties, online control without exact prior information would inevitably incur performance loss (*a.k.a. regret*). Quantifying such regrets can provide system designers with a better understanding of their design space.

In this paper, we address the above challenges by adopting an effective combination of Lyapunov optimization [18] and bandit learning [19]. More specifically, we make the following contributions.

- ◇ **Modeling and Formulation:** We formulate the problem of joint switch-controller association and control devolution with unknown system dynamics as a combinatorial multi-armed bandit (CMAB) problem with long-term queue stability constraints. For online control, we aim to reduce the time-average communication costs and computation and ensure the long-term stability of all queue backlogs. For online learning, we aim to minimize regret due to decision making under uncertainty.
- ◇ **Algorithm Design:** By exploiting the unique problem structure, we devise an effective Learning-Aided Switch-Controller Association and Control Devolution (*LASAC*) scheme, which achieves a proper integration of online control and online learning. Specifically, regarding online control, we adopt optimal control techniques [18] to handle the long-term queue stability constraints and conduct efficient online decision-making to minimize the total system costs. To achieve effective online learning, we employ *UCB1-tuned* [19], an improved version of classic upper-confidence bound (UCB1) method, to balance the exploration-exploitation tradeoff.
- ◇ **Performance Analysis:** Our theoretical analysis shows that LASAC can perform effective online control by achieving an $[O(V), O(1/V)]$ tradeoff between queue stability and system cost reduction with a tunable positive parameter V, and realize efficient learning by limiting the regret within a sub-linear $O(\sqrt{\log T/T})$ bound over a finite time horizon T.
- ◇ **Experimental Verification and Evaluation:** We conduct extensive simulations to evaluate LASAC and its variants. Results from our simulations not only verify the theoretical tradeoffs of LASAC but also show that LASAC and its variants effectively reduce total system costs with mild-value of regrets and queue stability guarantee.

The rest of this paper is organized as follows. We discuss related works in Section II. Then we present our system model and problem formulation in Section III. In Section IV, we demonstrate the design of LASAC, followed by its performance analysis. Next, we analyze our simulation results in Section V. Finally, we conclude this paper in Section VI. Note that all proofs are relegated to our technical report [20].

II. Related Work

Optimizations for SDN Systems: So far, a number of existing works [21] have been conducted to optimize the performance (*e.g.*, the control latency [22]–[24], resiliency [25]–[27], cost efficiency [25], *etc.*) of SDN systems. Among such works, there are two lines of research in recent years that focus on optimizing the effective control and optimization for switch-controller association [3]–[12] and control devolution [13]–[16], respectively. Instead of studying these two topics separately, later works [17] [28] considered the problem of joint switch-controller association and control devolution, then proposed *online* control and predictive control schemes to optimize system costs with queue stability guarantee, respectively. Although problems studied by such works are similar to our work, we would like to point out that all such works implicitly assume the *full* availability of instant system dynamics, which is hard to attain in practice. In contrast, our work assumes only *partial* availability of instant system dynamics (*e.g.*, queue backlog sizes of controllers but not system costs) upon decision making. Generalizing to such more settings brings great challenges and complexities to the scheme design, which are fully addressed by this paper.

Learning-aided Control for SDN Systems: During the past decade, there has been a growing interest in leveraging machine learning techniques to characterize and improve performances of SDN systems [29] such as those for routing optimization [30] [31], traffic classification [32] [33], and resource management [34]–[36]. Although the effectiveness of such works have been well justified, most of them were conducted based on *offline* learning techniques with readily available datasets, while *online* learning techniques have been considered by only few works. For example, Rehmani *et al.* [37] focused on SDN-based smart grids and proposed an approach based on ϵ-greedy method to learn link failure statistics and direct controllers to adapt switches' coordination to dynamic network conditions. In comparison, our work studies the problem of joint switch-controller association and control devolution with unknown system dynamics in SDN systems. We not only devise an effective learning-aided scheme with performance guarantee but also investigate the interplay between online control and online learning through theoretical analysis. To our best knowledge, this paper provides the first study on the joint control and learning for SDN systems.

Learning-aided Control under Bandit Settings: To date, the multi-armed bandit (MAB) model [38] has been extensively adopted to study sequential decision-making problems under uncertainty within a wide range of scenarios [39], such as the cache placement in wireless caching systems [40] [41], channel allocation in fog systems [42], collaborative filtering in recommendation systems [43], and beam tracking in millimeter-wave communication systems [44]. Among varieties of MAB settings, the most related to our work is the combinatorial multi-armed bandit model in [45]. In their model, each arm is assigned with a unique virtual queue to trace the proportion of time being selected, which serves as an indicator to ensure fairness constraints. Each queue backlog is only affected by the selection of its associated arm. Different from their setting, in our model, each queue backlog is not uniquely assigned to a particular arm. Instead, the change of queue backlogs during each round may result from the selection of multiple arms. Therefore, our model is more complicated and the resulting problem is more challenging to solve. Faced with such difficulties, we not only devise an effective scheme that solves the problem with a sublinear regret bound but also investigate the interplay between online control and online learning. Our results provide interesting insights for the study of learning-aided approaches.

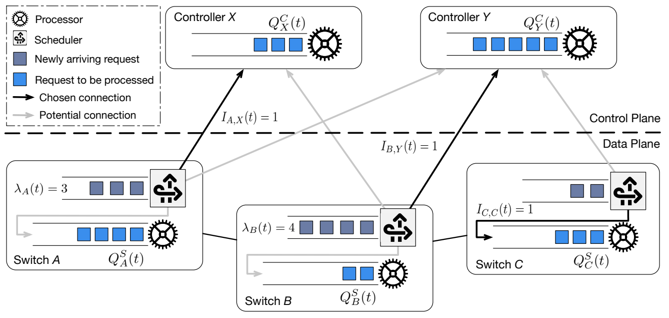

Fig. 1. An instance of our system model in time slot t, with three switches in the data plane and two controllers in the control plane. Each switch i generates a number of $\lambda_i(t)$ new requests and the scheduler makes decision $I_{i,\cdot}(t)$ to keep the new requests processed locally (*e.g.*, $I_{C,C}(t) = 1$), or forward them to one of its potentially associated controllers (*e.g.*, $I_{A,X}(t) = 1$). New requests are then appended to the corresponding queue backlogs and processed in a first-in-first-out manner.

III. Problem Formulation

A. Basic Model

We consider an SDN system that operates over a finite number of T time slots, as demonstrated in Figure 1. Each time slot is indexed by $t \in \{0, 1, 2, \ldots, T-1\}$. The control plane and data plane of the system are formed by a set of controllers (denoted by set $\mathcal{C}$) and a set of switches (denoted by set $\mathcal{S}$), respectively. Each switch $i \in \mathcal{S}$ has potential associations to a subset $\mathcal{C}_i \subseteq \mathcal{C}$ of controllers.

Due to the temporal variation of SDN system dynamics, the set of accessible controllers for each switch i may change over time. We use $A_i(t)$ to denote the accessible controller set of switch i during time slot t. By defining set $\mathcal{P}(\mathcal{C}_i)$ as the power set of $\mathcal{C}_i$, we assume that $p(Z_i) = \Pr\{A_i(t) = Z_i\}$ for each $Z_i \in \mathcal{P}(\mathcal{C}_i)$. Moreover, each $A_i(t)$ is assumed *i.i.d.* across time slots and independent across switches.

Next, for all requests in the system, we assume that they are homogeneous and each request can be *processed locally* or *uploaded* to its switch's potentially associated controllers.[1]

B. System Workflow

The overall workflow of the system proceeds as follows. At the beginning of each time slot t, each switch i generates a number $\lambda_i(t)$ of new requests, *e.g.*, *flow install request* [1]. The number $\lambda_i(t)$ is assumed to be upper bounded by some constant $\lambda_{\max}$. Then switch i should make its *devolution decision* about whether to process such new requests locally or not. If deciding to upload new requests to its potentially associated controllers, then switch i should make its *association decision* about which controller to associate with. With decisions being made, some switches upload new requests to controllers while the others process new requests locally. Meanwhile, switches and controllers process as many received requests as possible. Such a process is repeated during every time slot.

[1]In real systems, however, some requests may be heterogeneous and cannot be processed locally on switches due to limited information and functionality on switches or relevant statefulness on controllers. For such cases, our setting can be extended by imposing extra constraints on the scheduling decisions.

During such a process, owing to recently developed high-speed transmission techniques [46], we assume that the transmission latencies between switches and controllers are much smaller than each time slot's length. Besides, due to service capacity limits, switches and controllers may not be able to finish all new requests within one time slot. To cope with the untreated requests, each switch (or controller) also maintains a processing queue to buffer them. In the following subsections, we show how we model the scheduling decisions of switches and controllers and their queueing dynamics in detail.

C. Scheduling Decisions

For each switch i, we denote its *devolution decision* by variable $I_{i,i}(t) \in \{0, 1\}$. If $I_{i,i}(t) = 0$, then switch i will associate with one of its potentially associated controllers; otherwise, it will append new requests to its own queue and process them locally. In the meantime, we denote switch i's *association decision* by variables $\{I_{i,j}(t)\}_{j \in \mathcal{C}_i}$. Specifically, $I_{i,j}(t) = 1$ indicates that controller j is chosen to process new requests and zero otherwise.[2] For simplicity, we use $\mathbf{I}_i(t) \triangleq \{I_{i,i}(t)\} \cup \{I_{i,j}(t)\}_{j \in \mathcal{C}}$ to denote the decisions of switch i in time slot t. We summarize the decisions of all switches by $\mathbf{I}(t) \triangleq \{\mathbf{I}_i(t)\}_{i \in \mathcal{S}}$.

Considering the high costs of simultaneous communication with multiple controllers [47], each switch i is restricted to choose at most one controller in each time slot. Then we have

$$I_{i,i}(t) + \sum_{j \in \mathcal{C}_i} I_{i,j}(t) = 1, \ \forall\, i \in \mathcal{S}. \tag{1}$$

Besides, recall that $A_i(t)$ denotes the set of accessible controllers of switch i during time slot t. Then we have

$$I_{i,j}(t) = 0, \ \forall j \in \mathcal{C} \backslash A_i(t). \tag{2}$$

D. Queueing Dynamics

For each switch i, we define $Q_i^S(t)$ as its queue backlog size at the beginning of time slot t. Correspondingly, for each controller j, we define $Q_j^C(t)$ as its queue backlog size at the beginning of time slot t. Note that due to the homogeneity of requests, each backlog size is equal to the number of requests stored in the queue.

Based on the workflow described in Subsection III-A, the queueing dynamics in the system can be characterized by the following equations. For each switch $i \in \mathcal{S}$,

$$Q_i^S(t+1) = \left[Q_i^S(t) + I_{i,i}(t)\lambda_i(t) - \mu_i^S(t) \right]^+, \tag{3}$$

where we define operator $[\cdot]^+ \triangleq \max\{0, \cdot\}$. Next, for each controller $j \in \mathcal{C}$,

$$Q_j^C(t+1) = \left[Q_j^C(t) + \sum_{i \in \mathcal{S}} I_{i,j}(t)\lambda_i(t) - \mu_j^C(t) \right]^+. \tag{4}$$

E. Optimization Objectives:

On the one hand, the benefit from uploading requests lies in that controllers have adequate service capacities and thus they

[2]In practice, depending on requests' particular states, the decision making can be adapted such that during each time slot, part of new requests get uploaded and others stay locally.

incur shorter processing latencies than switches. Such a benefit may be offset unexpectedly by considerable communication costs (*e.g.*, round-trip time [17]) of request uploading. On the other hand, if most requests are kept processed locally, switches may be overwhelmed by prohibitively high computational costs. As a result, the decision-making for joint switch-controller association and control devolution should carefully manage the balance between computation and communication costs. Below we specify the definitions of and the constraints on such costs in our model.

Communication Cost: For each switch i and each of its accessible controllers $j \in A_i(t)$, we denote their communication cost of sending a request during time slot t by $W_{i,j}(t)$ ($\leq w_{\max}$ for some positive constant $w_{\max}$). Such communication costs are assumed *i.i.d.* across time slots with a finite expectation $\mathbb{E}\{W_{i,j}(t)\} = \bar{w}_{i,j}$. Then with decisions $\mathbf{I}(t)$, the total communication cost in time slot t is given by

$$W(t) \triangleq \hat{W}(\mathbf{I}(t)) = \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{C}} I_{i,j}(t)\lambda_i(t)W_{i,j}(t). \tag{5}$$

Computational Cost: For each switch i, we denote its computational cost of processing each request locally during time slot t by $M_i(t)$ ($\leq m_{\max}$ for some positive constant $m_{\max}$). Then its total computational cost in time slot t is given by

$$M(t) \triangleq \hat{M}(\mathbf{I}(t)) = \sum_{i\in\mathcal{S}} I_{i,i}(t)\lambda_i(t)M_i(t). \tag{6}$$

Queue Stability: Besides system costs, the stability of queue backlogs in the system is also worth considering. Intuitively, achieving queue stability means that no queue backlog on switches or controllers would increase infinitely and eventually lead to system disruption. It also ensures that requests would not queue up intensively on any switch or controller, which by *Little's law* [48] implies that no requests will ever experience excessively long queueing delay. In this paper, we define queue stability [18] as

$$\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\Big\{\sum_{i\in\mathcal{S}}Q_i^S(t)+\sum_{j\in\mathcal{C}}Q_j^C(t)\Big\} < \infty. \tag{7}$$

F. Problem Formulation:

In our model, we aim to minimize the time-average expectation of the total costs of communication and computation over a finite time horizon T, subject to long-term queue stability. Our problem formulation is given as follows.

$$\begin{aligned}\underset{\{\mathbf{I}(t)\}_{t=0}^{T-1}}{\text{Minimize}} \quad & \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\{W(t)+M(t)\right\}\\ \text{Subject to} \quad & (1), (2), (7).\end{aligned} \tag{8}$$

IV. Algorithm Design and Performance Analysis

Provided that the knowledge of system dynamics is fully known *a priori*, Problem (8) can be solved asymptotically optimally in the long run by applying Lyapunov optimization techniques [18]. However, such knowledge is generally hard to attain in practice. For example, the communication and computational costs on switches often remain unknown at the beginning of each time slot. Instead, they can be revealed or inferred from feedback information after requests being delivered or processed. Besides, the statistics of each switch i's accessibility to controllers $A_i(t)$ are also unknown to the systems. Faced with such uncertainties, online learning is needed to aid the online control procedure. To this end, we have the following challenges to be addressed.

The *first* challenge relates to the online learning procedure. To minimize system costs under such uncertainties, the designed scheme needs to keep a decent balance between *exploration* and *exploitation*. Exploration, *i.e.*, favoring those rarely chosen queue backlogs with uncertain costs, allows switches to learn as much about the costs of different choices as possible. However, too much exploration may incur unexpectedly higher costs and lead to performance loss (*a.k.a* regret). In contrast, if switches stick to exploitation, *i.e.*, selecting queues with the empirically lowest costs, they may miss potentially better choices and incur undesired regrets.

The *second* challenge relates to the online control procedure. In particular, the design must carefully handle the non-trivial tradeoff between system cost reduction and queue stability. This is because improperly choosing the queue backlogs with empirically lowest costs may overwhelm the corresponding switches/controllers. Moreover, maintaining the tradeoff through a series of online decision making makes the design even more complicated.

The *third* challenge roots in the interplay between online learning and online control. On the one hand, recall that the aim of introducing online learning is to reduce uncertainties of system dynamics and to facilitate the online control procedure. However, if conducted improperly, online learning may misguide the online control procedure and cause more regrets. On the other hand, online control aims to make the best use of available information to conduct effective decision making with instructive feedback information. But if poorly designed, online control may incur noisy feedback to online learning and impose adverse effects on learning efficiency.

Given the above challenges, a carefully integrated design of online control and online learning is needed. In fact, we can decompose Problem (8) into a series of matching subproblems over time slots. During each time slot, a subset of connections is chosen between switches and their potential targets (including themselves and accessible controllers), subject to constraints (1) and (2). Each selected connection has a time-varying weight with a constant mean. The goal is to minimize the time-average total weight over selected matchings across time slots. Faced with such an online decision-making problem under uncertainty, we switch to investigating Problem (8) from the perspective of combinatorial multi-armed bandit (CMAB) with sleeping arms [45]. Below we first introduce the background and settings of multi-armed bandit to motivate our reformulation. Then we reformulate Problem (8) as a CMAB problem with long-term constraints, followed by our algorithm design and performance analysis.

A. Motivation for Reformulation

The multi-armed bandit (MAB) model [38] has been extensively adopted to study a wide range of sequential decision-making problems under uncertainty [39]. Within the canonical settings of MAB, a player is considered to play a multi-armed bandit over a finite number of rounds. During each round, the player pulls one of the arms and receives a reward that is sampled from some distribution (*w.r.t.* the selected arm) with an unknown mean. The objective of the model is to devise an online policy for the player to maximize the cumulative reward from its successive plays. Such a model is well suited to scenarios in which the decision maker (the player) picks a single choice for each round of decision making. Back to our model, a direct idea is that we can view each switch as a distinct player and its accessible controllers as the arms. However, such a reformulation does not fit our problem settings due to the following reasons.

First, the decision making of each switch is not independent of each other. Specifically, each controller is often associated with multiple switches during each time slot (round). Accordingly, the decisions of such switches will jointly change the controller's queue backlog and influence their subsequent decision making. Second, unlike the settings of MAB model, the set of arms for the decision maker keeps changing across different time slots. Therefore, faced with the coupled nature of decision making among switches and time-varying availability of switch-controller connections, the basic MAB model does not apply to our problem settings.

Instead of considering the problem from the perspective of individual switches, we view the whole data plane as a player and each switch-controller connection as a distinct arm. Then the problem becomes how to pick a set of arms (instead of one arm) for the player during each time slot to maximize its cumulative rewards (*i.e.*, equivalent to the minus of its incurred costs) over time slots. A similar setting has been studied by [45] as the combinatorial multi-armed bandit (CMAB) model with sleeping arms. Following such an idea, in the subsequent subsections, we formalize the settings of CMAB and then demonstrate our problem reformulation.

B. Combinatorial Bandit Settings

Suppose that a player plays a bandit with N arms over T time slots. The set of all arms is denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$. During each round t, only a subset of arms are available to be played by the player. We denote the set of such arms by $A(t) \in \mathcal{P}(\mathcal{N})$. The availability of arms is assumed to follow a fixed but unknown distribution $p(Z) = \Pr\{A(t) = Z\}$ for $Z \in \mathcal{P}(\mathcal{N})$. Then given $A(t)$, the player should choose and pull a subset of arms $f(t)$ (*a.k.a. super arm*) whose size is no greater than some constant m. Accordingly, we define $\mathcal{F}(A(t)) \triangleq \{f \subseteq A(t) : |f| \le m\}$ as the set of feasible super arms. For each arm $n \in f(t)$, it returns some reward $X_n(t)$ to the player which follows a fixed but unknown distribution with a mean $\bar{x}_n$. Therefore, after pulling the arms in super arm $f(t)$, the player will receive a compound reward of $R(t) \triangleq \sum_{n \in f(t)} X_n(t)$. The goal of the player is to find a policy π that maximizes the expected time-average compound reward (denoted by $\mathbb{E}\big\{\frac{1}{T}\sum_{t=0}^{T-1} R(t)\big\}$) over T rounds. Such a goal is also equivalent to minimizing reward loss (*regret*) due to decision making under uncertainty. For each policy π, its regret is characterized by the difference between the maximum achievable reward (when full knowledge of system dynamics is given) and its time-average expected reward under; *i.e.*,

$$R_\pi(T) \triangleq R^* - \mathbb{E}_\pi\Big\{\frac{1}{T}\sum_{t=0}^{T-1}\sum_{n \in f(t)} X_n(t)\Big\}, \tag{9}$$

where $R^* \triangleq \sum_{Z \in \mathcal{P}(\mathcal{N})} p(Z) \sum_{f \in \mathcal{F}(Z)} q_Z^*(f) \sum_{n \in f} \bar{x}_n$ and $q_Z^*(f)$ denotes the optimal expected time fraction of choosing super arm f from set $\mathcal{P}(Z)$. In the following subsection, we fit our model and problem formulation into the above settings.

C. Problem Reformulation

We regard the data plane as the player. Then we view each potential connection between switches and controllers as an arm. Accordingly, the arm set $\mathcal{N}$ is given by $\mathcal{N} \triangleq \{(i,i)\}_{i \in \mathcal{S}} \cup \{(i,j)\}_{i \in \mathcal{S}, j \in \mathcal{C}_i}$. For each arm $(i,i) \in \mathcal{N}$ with $i \in \mathcal{S}$, pulling it corresponds to the decision to make switch i process new requests locally; for each arm $(i,j) \in \mathcal{N}$ with $i \in \mathcal{S}$ and $j \in \mathcal{C}_i$, pulling it corresponds to the decision to associate switch i with controller j. Then we denote the set of all available arms during time slot t by $A(t) \triangleq \{(i,i)\}_{i \in \mathcal{S}} \cup \big(\bigcup_{i \in \mathcal{S}} A_i(t)\big)$. Note that such an available arm set is also *i.i.d.* across time slots and independent across switches. Then the probability of attaining such an arm set by $p(A(t)) = \prod_{i \in \mathcal{S}} p(A_i(t))$. Accordingly, we define the set of feasible super arms $\mathcal{F}(A(t))$ as

$$\begin{aligned}&\mathcal{F}(A(t)) \triangleq \\ &\{f \subseteq A(t) : \mathbb{1}_{(i,i) \in f} + \sum_{j \in \mathcal{C}_i} \mathbb{1}_{(i,j) \in f} = 1,\ \forall\, i \in \mathcal{S}\},\end{aligned} \tag{10}$$

where $\mathbb{1}_{n \in f}$ indicates whether arm n is contained in super arm f. Note that (10) ensures that each super arm will contain exactly $|\mathcal{S}|$ arms. In other words, given the accessible controller set $A(t)$, the constraint in (10) is equivalent to constraint (1).

Next, we define each arm (i,i)'s reward during time slot t as $X_{i,i}(t) = -M_i(t)$, *i.e.*, the negative of the computational cost of switch i. For each arm (i,j), we use $X_{i,j}(t) = -W_{i,j}(t)$ to denote the reward of associating switch i with controller j. Given any super arm $f(t)$, its corresponding compound reward $R(t)$ is given by

$$R(t) \triangleq \sum_{(i,k) \in f(t)} X_{i,k}(t). \tag{11}$$

Defined in the above way, minimizing the time-average total system costs in (8) is equivalent to maximizing the cumulative rewards over T time slots. Thus, Problem (8) can be rewritten as follows

$$\begin{aligned}\underset{\{f(t) \in \mathcal{F}(A(t))\}_{t=0}^{T-1}}{\text{Maximize}} \quad & \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\Big\{\sum_{(i,k) \in f(t)} X_{i,k}(t)\Big\} \\ \text{Subject to} \quad & (7).\end{aligned} \tag{12}$$

Remark: Problem (12) is more complicated than the original CMAB problem. Due to constraint (7), the decision-making process should not only maximize the cumulative compound reward but also maintain queue stability on switches and controllers in the long run. This requires the online control procedure not to narrow down its sight onto the arms with the empirically highest rewards; instead, it should also switch amongst multiple arms by taking their queue backlog sizes into account. Note that in [45], queue stability is considered as a form of fairness guarantee. Meanwhile, in their model, different arms' queue backlogs are independent of each other. However, in our model, each arm corresponds to a switch-controller connection that directs requests sent from a switch to one of its accessible controllers or itself. Accordingly, each controller's queue is shared by multiple switches (equivalently by multiple arms). Such a coupling among arms makes our problem even more complicated.

D. Algorithm Design

Regarding online learning, we adopt *UCB1-tuned* [19] to estimate each arm's quality based on its feedback reward information. As an extended version of classic UCB1 (upper-confidence-bound1) method [49], UCB1-tuned makes further use of feedback information to estimate the reward variance of each arm and reconcile the *exploration-exploitation* tradeoff. Owing to such fine-grained evaluations, UCB1-tuned has been shown to outperform various bandit learning methods [19].

In particular, for each arm (i,k), we define $h_{i,k}(t)$ as the number of times it has been chosen by the end of time slot t. Formally, we write $h_{i,k}(t) \triangleq \sum_{\tau=1}^{t} I_{i,k}(\tau)$ and assume $h_{i,k}(-1)=0$. Then the sample mean of rewards is given by

$$\hat{x}_{i,k}(t) \triangleq \frac{\sum_{\tau=1}^{t} X_{i,k}(\tau) I_{i,k}(\tau)}{h_{i,k}(t)}. \tag{13}$$

Note that if arm (i,k) has never been chosen before, we set $\hat{x}_{i,k}(t)=0$.

Next, we define the UCB1-tuned estimate for arm (i,k)'s mean reward as [3]

$$\tilde{x}_{i,k}(t+1) \triangleq \min\left\{u_{i,k}(t), 0\right\}, \tag{14}$$

where the upper confidence bound $u_{i,k}(t)$ is defined as

$$u_{i,k}(t) \triangleq \hat{x}_{i,k}(t) + \beta \sqrt{\frac{\log(t+1)}{h_{i,k}(t)} \cdot \min\{1/4, V_{i,k}(h_{i,k}(t))\}}, \tag{15}$$

and operator $\min\{\cdot, 0\}$ is to ensure the non-positiveness of reward estimates, since system costs are always non-negative. In (15), parameter β measures the relative weight of uncertainty credit (the second term in (15), *a.k.a.* the exploration term) to the empirical estimate of mean reward $\hat{x}_{i,k}(t)$ (*a.k.a.* the exploitation term). Accordingly, different values of parameter β reflect different degrees of exploration-exploitation tradeoff. Note that we can view the exploration term as the scaled error margin of the reward estimate. Specifically, in the exploration term, $V_{i,k}$ is defined as

$$V_{i,k}(h_{i,k}(t)) \triangleq \Big(\frac{1}{h_{i,k}(t)} \sum_{\tau=1}^{h_{i,k}(t)} [X_{i,k}(\tau)]^2 - \bar{X}_{i,k}^2 + \sqrt{\frac{2\log(t+1)}{h_{i,k}(t)}}\Big), \tag{16}$$

which denotes an optimistic empirical estimate of the reward variance by pulling arm (i,k).

[3] $\tilde{x}_{i,k}(-1)$ is initialized as zero for all $i \in \mathcal{S}$ and $k \in \{i\} \cup \mathcal{C}_i$.

Regarding online control, we adopt Lyapunov drift techniques [18] to cope with the tradeoff between system cost minimization and queue stability. The key idea is to pull arms greedily based on their instant queue backlog sizes and empirical reward estimates during each time slot. Through a series of such online decision making, cumulative reward maximization and long-term queue stability can be properly balanced. We relegate the detail to Subsection III-D.

By carefully integrating online learning with online control, we devise an effective Learning-Aided Switch-controller Association and Control Devolution scheme (*LASAC*). We show its pseudo-code in Algorithm 1 with remarks as follows.

Algorithm 1 Learning-Aided Switch-controller Association and Control Devolution (LASAC)

```
Input: At the beginning of each time slot $t$, given backlog
    sizes $\boldsymbol{Q}(t)$ and the set of accessible controllers $A_i(t)$ for
    each switch $i$.
Output: A series of decisions $\{\boldsymbol{I}(t)\}_t$ over time horizon $T$.
 1: for each time slot $t \in \{0, 1, \dots, T-1\}$:
 2:   for each switch $i \in \mathcal{S}$:
 3:     for each candidate $k \in \{i\} \cup \mathcal{C}_i$:
 4:       if $h_{i,k}(t-1) > 0$ then:
 5:         Update $\tilde{x}_{i,k}(t)$ according to (14).
 6:       else
 7:         Set $\tilde{x}_{i,k}(t) \leftarrow 0$.
 8:     Select candidate $k^*$ such that
              $k^* \in \arg\min_{k \in \{i\} \cup A_i(t)} l_{i,k}(t)$,
        where $l_{i,k}(t)$ is defined as
              $l_{i,k}(t) \triangleq \begin{cases} Q_i^S(t) - V \cdot \tilde{x}_{i,i}(t) & \text{if } k = i, \\ Q_k^C(t) - V \cdot \tilde{x}_{i,k}(t) & \text{otherwise.} \end{cases}$   (17)
 9:     Set $I_{i,k^*}(t) \leftarrow 1$ and $I_{i,k}(t) \leftarrow 0$ for $k \neq k^*$.
10:     if $k^* = i$ then
11:       Append new requests to switch $i$'s local queue.
12:     else
13:       Forward new requests to controller $k^*$.
14:     Collect the corresponding reward $X_{i,k^*}(t)$.
15:     Update $h_{i,k}(t)$ and $\hat{x}_{i,k}(t)$ for all $k$.
16:   Update all queue backlogs according to (3) and (4).
```

Remark 1: By maintaining the estimate (15) for each arm in every time slot, LASAC well balances the tradeoff between exploration and exploitation for online learning. Particularly, if an arm has been pulled for a sufficiently large number of times, the exploitation term will become dominant and the estimate will count more on the empirical mean. Otherwise, the exploration term comes into play. Recall that $V_{i,k}(h_{i,k}(t))$ is an optimistic estimate of the variance of the arm's reward. Then intuitively, under UCB1-tuned, the credit of uncertainty for pulling each arm is large not only when the arm is explored insufficiently but also when its reward variance estimate is high. In this way, exploration is further encouraged in LASAC, which conduces to learning efficiency.

Remark 2: According to line 8 in Algorithm 1, LASAC manages the tradeoff between cumulative compound reward

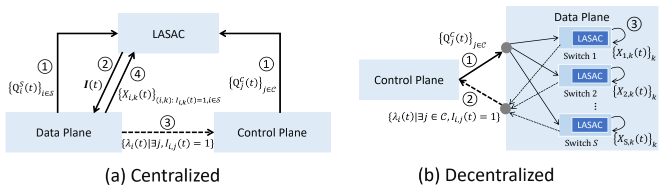
LASAC
①
$\{Q_i^S(t)\}_{i\in\mathcal{S}}$
②
$\boldsymbol{I}(t)$
④
$\{X_{i,k}(t)\}_{(i,k):\, I_{i,k}(t)=1, i\in\mathcal{S}}$
①
$\{Q_j^C(t)\}_{j\in\mathcal{C}}$
Data Plane
③
Control Plane
$\{\lambda_i(t) | \exists j, I_{i,j}(t)=1\}$
(a) Centralized
Data Plane
③
$\{Q_j^C(t)\}_{j\in\mathcal{C}}$
①
Control Plane
②
$\{\lambda_i(t) | \exists j \in \mathcal{C}, I_{i,j}(t)=1\}$
LASAC
Switch 1
$\{X_{1,k}(t)\}_k$
LASAC
Switch 2
$\{X_{2,k}(t)\}_k$
LASAC
Switch S
$\{X_{S,k}(t)\}_k$
(b) Decentralized

Fig. 2. Two implementations of LASAC

maximization and queue stability by jointly considering each arm's reward estimate and its associated instant queue backlog size. By (17), LASAC takes instant queue backlog sizes as the indicator of queue stability. Meanwhile, LASAC uses the value of parameter V to determine the relative importance of reward maximization compared to queue stability. The larger the value of V is, the more willing LASAC is to pick the arm with the empirically highest reward (lowest cost) for each switch. In contrast, if the value of parameter V is small, then LASAC will favor those arms with small queue backlog sizes. In practice, the value of parameter V can be tuned around the ratio of the magnitude of individual queue backlog capacity to that of system costs.

Remark 3: LASAC can run in a distributed manner with a computational complexity of $O(|\mathcal{C}|)$. Specifically, given instant queue backlog sizes and accessible states of its accessible controllers, each switch can conduct its own decision making independently. In practice, LASAC can be implemented and deployed in either a centralized or a decentralized fashion, which are visualized in Figures 2(a) and 2(b), respectively.

Under centralized implementation, LASAC is deployed on a particular server or cluster which is independent of the control plane and the data plane. At runtime, LASAC first collects instant system dynamics including queue backlog sizes on switches and controllers to conduct the decision making. Then it will spread such decisions onto switches, so that switches can schedule their requests accordingly. Finally, the incurred system costs are measured and sent back to LASAC. The advantage of such an implementation is that it requires no modification on switches; *i.e.*, all necessary information for decision making can be obtained via standard OpenFlow APIs [50]. This is well-suited for scenarios with a large-scale data plane in which switches' compute resources are scarce. However, such an implementation may also be a single point of failure and the performance bottleneck for a large-scale data plane. Besides, it requires extra communication overheads for exchanging information between the system and LASAC.

Under decentralized implementation, LASAC is implemented as a function module and deployed on each switch. Then at runtime, each switch periodically updates its information about system costs and queue backlogs from the control plane, to conduct its decision making independently. Although such an implementation requires modifications and incurs extra computations on switches, compared to the centralized implementation, it requires less amounts of information exchange and incur lower communication overheads. Moreover, the resulting distributed decision making also conduces to better scalability and fault tolerance.

E. Performance Analysis

In general, we have two questions about the performance of LASAC. One is that, with learning and control procedures being tightly coupled, can LASAC guarantee queue stability in the long run? The other is that in the face of various uncertainties, what is the regret upper bound that LASAC can achieve? How would different factors such as the length of time horizon T affect the regret bound? These two questions are answered by the following theorems, respectively.

Queue Stability: For any mean service rate vector $\boldsymbol{\mu} = (\bar{\mu}_1^S, \ldots, \bar{\mu}_{|\mathcal{S}|}^S, \bar{\mu}_1^C, \ldots, \bar{\mu}_{|\mathcal{C}|}^C)$, it is said to be *feasible* if there exists such a scheduling scheme that its decision making ensures that the mean arrival rate is no greater than the mean service rate for each switch and controller in the system. We define the set of all such feasible mean service rate vectors as the *maximal feasibility region*. We have the following result.

Theorem 1: Provided that the mean service rate vector $\boldsymbol{\mu}$ lies in the interior of the maximal feasibility region, then LASAC can achieve queue stability (7) in the systems. In other words, there exist positive constants B and ϵ such that

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left\{ \sum_{i\in\mathcal{S}} Q_i^S(t) + \sum_{j\in\mathcal{C}} Q_j^C(t) \right\} \le \frac{B}{\epsilon} + V \cdot |\mathcal{S}| \cdot \lambda_{\max} \cdot \max\left\{w_{\max}, m_{\max}\right\}. \tag{18}$$

Regret Bound: The following theorem gives an upper bound for the regret of LASAC over time horizon T.

Theorem 2: Over T time slots, the time-average regret (9) of LASAC has an $O(\sqrt{\log T/T})$ sub-linear upper bound; i.e., there exists some positive constant $\tilde{B}$ such that

$$R_{LASAC}(T) \le \frac{\tilde{B}}{V} + 2|S| \cdot (|C|+1) \cdot \Big[\beta\sqrt{\frac{\log T}{T}} + \frac{1}{T}(G_\beta + \frac{1}{2}) \cdot \max\{w_{max}, m_{max}\}\Big], \tag{19}$$

where $G_\beta \triangleq \sum_{t=1}^{\infty} t^{-\frac{\beta^2}{2}}$ is a function of parameter β.

Discussion: From Theorems 1 and 2, we know that:

1) Regarding online control, LASAC can achieve a tunable $[O(1/V), O(V)]$ tradeoff between system cost reduction (reward maximization) and queue stability in terms of an upper bound for the total queue backlog size. Particularly, according to (18), the upper bound of queue backlog size is linear in the value of V; and according to (19), the regret is inversely proportional to the value of V. That being said, larger values of V lead to larger total queue backlog sizes and smaller regrets, whilst smaller values of V result in smaller backlog sizes and larger regrets.
2) Regarding online learning, LASAC can achieve different levels of exploration-exploitation tradeoff with different values of β. Parameter β appears in two terms of (19), *i.e.*, $\beta\sqrt{\log T/T}$ and $(G_\beta + 1/2)\max\{w_{\max}, m_{\max}\}/T$. Recall from (15) that the greater the value of parameter β, the more explorations LASAC conducts. However, the first term in (19) suggests that more explorations incur a higher regret (equivalently high system costs). Nonetheless, the regret will decrease over time at a rate of $O(\sqrt{\log T/T})$. Besides, if the value of parameter β

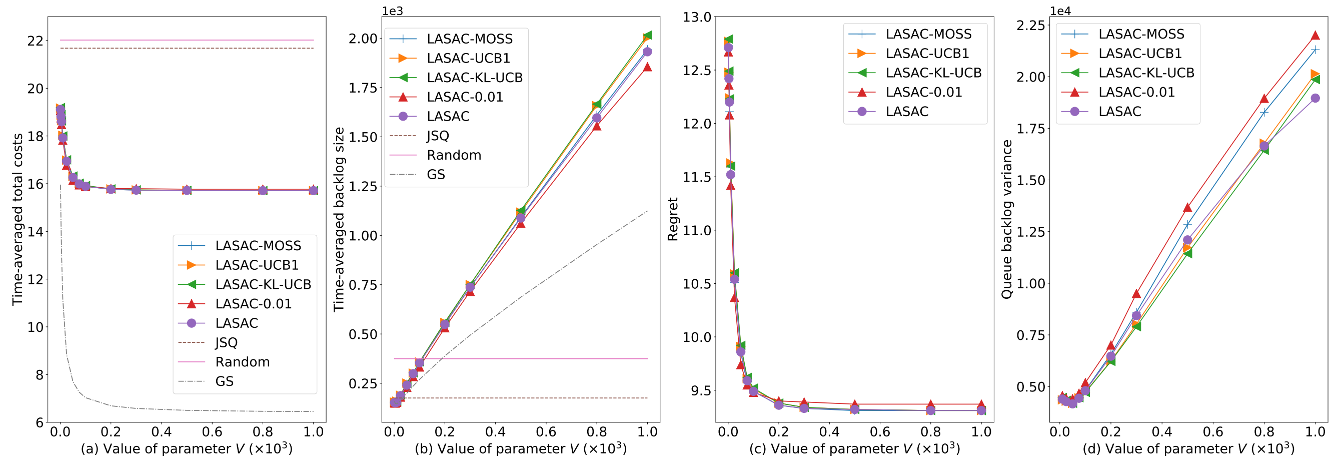

Fig. 3. Performance Comparison under Different Values of V with $\beta = 2$

is tuned too small, then the series G_β may not converge and hence the bound in (19) becomes indefinite. This reflects the downside of over-conservative exploration. That is, each switch may blindly choose the controllers with the empirically lowest cost estimates and miss other potentially better candidates.

3) The first and second terms in (19) also quantify the impact of online control and online learning on the regret, respectively. However, there are other subtle interactions between control and learning. For example, the online control procedure would also implicitly enforce exploration, which is discussed in Section V.

V. Simulation

A. Basic Settings

We conduct simulations in an SDN system that evolves over 5×10^6 time slots. Each time slot has a length of $10ms$. The system contains 10 switches in the data plane and 4 controllers in the control plane. Each switch has three potentially connected controllers, with each controller being chosen independently randomly with a probability ranging from 0.8 to 1. Meanwhile, for each switch and controller, we set their mean processing capacities as 2 and 12 requests per time slot, respectively. The mean computation and communication costs of each switch are set equal to the number of cores assigned to process requests and the number of hops to its associated controllers, respectively. Besides, request arrivals on each switch follow the distribution drawn from the measurement of real-world networks [51]. As for the value of parameter V, recall that by (17), it measures the tradeoff between reward maximization (system cost minimization) compared to queue stability. Accordingly, when cost minimization and queue stability are deemed equally important, the value of V is about the ratio of the magnitude of queue backlog size to that of system costs, which is 100 under our settings. In simulations, we adjust the value of V to investigate system performance under different levels of tradeoffs. To be specific, we take the value of parameter V from range $[1, 1000]$. To eliminate the impact of randomness, all results are averaged over 20 runs.

Baseline Schemes: We compare LASAC with the following baseline schemes which also proceed on a per-time-slot basis.

- ◇ Greedy Scheduling (GS) [17]: Each switch is assumed to have *full knowledge* about all communication and computational costs at the beginning of each time slot. It makes decisions with (17) based on the actual information rather than the empirical estimates.
- ◇ Random scheme: Each switch uniformly randomly sends requests to itself or one of its accessible controllers.
- ◇ Join-the-shortest-queue (JSQ): Each switch chooses the controller (or itself) with the smallest queue backlog size.

Besides, we also propose some variants of LASAC, with different ways to handle the exploration-exploitation tradeoff.

- ◇ LASAC-with-ϵ-Greedy (LASAC-ϵ): With probability ϵ, each switch selects one of its accessible controllers or itself uniformly randomly. Otherwise, the same decision making as LASAC is conducted.
- ◇ LASAC-X: Variants of this type follow the same decision making as LASAC, except that switches replace the reward estimate (15) by other UCB variant X, including UCB1 [49], MOSS [52], and KL-UCB [53].

B. Simulation Results and Analysis

We evaluate the performance of LASAC and its variants by examining how they handle the tradeoff between system cost reduction and queue stability. We also investigate the interplay between online learning and online control under different parameter settings.

Performance under different choices of parameter V: In Figures 3(a) - 3(d), we set $\beta = 2$ and evaluate different schemes as the value of V increases from 1 to 1000.[4]

Figures 3(a) and 3(b) show that GS achieves near-optimal total costs as the value of V increases to 1000. This is owing to its full knowledge about communication and computation costs during each time slot upon decision making. In contrast, without such prior information, LASAC and its variants

[4]Note that in (19), the series G_β converges only when $\beta \in (\sqrt{2}, +\infty)$. When $\beta \in [0, \sqrt{2}]$, the series G_β diverges and Theorem 2 holds trivially.

incur higher total costs and larger queue backlog sizes than GS. Meanwhile, Random and JSQ, although achieving more balanced workloads with small backlog sizes, achieve higher (up to 40.1%) costs than LASAC as they make no use of system cost information.

More specifically, as the value of V increases from 1 to 500, LASAC and its variants lead to about 21.6% reduction in system costs and eventually converge thereafter. The corresponding regrets of LASAC and its variants are shown in Figure 3(c). The cost reduction comes with linear growth in the total queue backlog size and, by *Little's theorem* [48], a longer request delay. As shown in Figure 3(d), the increase in the total queue backlog size is mainly due to the decisions made to reduce system costs during the online control procedure. For instance, some controllers have lower communication costs to more switches than other controllers. Accordingly, they are more likely to become the preferable choices of switches. Recall that by (17), the larger the value of V is, the more willing switches are to send new requests to such controllers. As a result, such controllers will be loaded with more requests since their service capacities are fixed across all simulations. This demonstrates the increase in the total backlog size. Such results verify our theoretical tradeoff between system cost reduction and queue stability in Section V-D. In practice, the value of V can be tuned to achieve both low costs and small queue backlog sizes (*e.g.*, $V \in [1, 100]$ in our simulations).

The interplay between parameters V and β: To investigate the interplay between online control and online learning, we evaluate the performance of LASAC under various combinations of parameters V and β. We vary their values in ranges $[10, 1000]$ and $[2, 1000]$, respectively. Besides, we also evaluate the performance of LASAC given $\beta = 0$, which by (15) corresponds to the strategy with pure exploitation. All results are shown in Figures 4(a) - 4(d).

First, we focus on the impact of parameter β on system performance. Figures 4(a) and 4(b) show that larger values of β generally lead to higher total system costs (correspondingly larger regrets, as shown in Figure 4(c)) and smaller total queue backlog sizes. Intuitively, according to (15), parameter β measures the balance between exploration and exploitation. The greater the value of β is, the more exploration LASAC intends to conduct. Therefore, a large value of β results in more frequent exploration and hence excessive system costs. In the meantime, frequent exploration prevents LASAC from sticking to forwarding requests to some particular controllers or processing them locally. As shown in Figure 4(d), this will lead to more balanced queue backlogs across controllers and switches, or equivalently, smaller queue backlog variance.

Interestingly, we find that when $\beta = 0$, LASAC performs as comparably well as in the cases with $\beta \in [2, 10]$. Intuitively, under such a policy with pure exploitation, switches would stick to choosing controllers with initially high empirical estimates and ignore other potentially better candidates, thereby incurring high system costs. However, owing to the interplay between online control and online learning, LASAC still maintains a balance between exploration and exploitation even when $\beta = 0$. In fact, LASAC may conduct pure exploitation initially. But as soon as those frequently chosen controllers

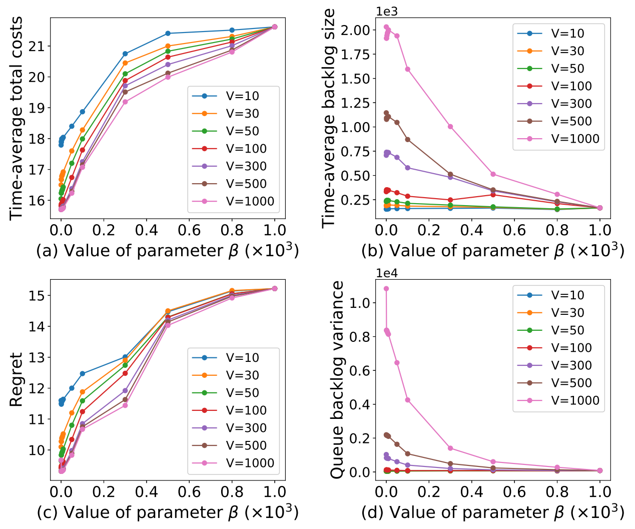

Fig. 4. Performance Comparison under Different Values of β

become overloaded due to pure exploitation, LASAC would have the switches turn to other controllers. In other words, exploration is implicitly enforced by the online control procedure to ensure queue stability.

From Figure 3 we know that larger values of V lead to lower system costs and larger total backlog sizes. Nevertheless, Figure 4 shows that as the value of parameter β increases, the impact of parameter V becomes gradually insignificant. Eventually when $\beta = 1000$, *i.e.*, LASAC is forced to make massive explorative decisions such that each switch frequently forwards new requests back and forth among controllers. As a result, the role of online control becomes marginalized. This demonstrates another interplay between online control and online learning. In practice, the choice of parameter β depends on the tradeoffs in system design.

VI. Conclusion

In this paper, we proposed the first scheme with an integrated design of online learning and online control to jointly conduct switch-controller association and control devolution in SDN systems. Our theoretical analysis showed that LASAC can achieve a tunable tradeoff between queue stability and system cost reduction while ensuring a sub-linear regret bound over a finite time horizon. We conducted extensive simulations to verify such theoretical results and evaluate the performance of LASAC and its variants.

References

[1] D. Kreutz, F. M. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.
[2] A. Gushchin, A. Walid, and A. Tang, "Scalable routing in sdn-enabled networks with consolidated middleboxes," in *Proceedings of ACM SIGCOMM HotMiddlebox*, 2015.
[3] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A distributed control platform for large-scale production networks." in *Proceedings of USENIX OSDI*, 2010.
[4] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of Internet Network Management Conference on Research on Enterprise Networking*, 2010.

[5] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an elastic distributed sdn controller," in *Proceedings of ACM HotSDN*, 2013.

[6] A. Krishnamurthy, S. P. Chandrabose, and A. Gember-Jacobson, "Pratyaastha: An efficient elastic distributed sdn control plane," in *Proceedings of ACM HotSDN*, 2014.

[7] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: state distribution trade-offs in software defined networks," in *Proceedings of ACM HotSDN*, 2012.

[8] T. Wang, F. Liu, J. Guo, and H. Xu, "Dynamic sdn controller assignment in data center networks: Stable matching with transfers," in *Proceedings of IEEE INFOCOM*, 2016.

[9] 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, 2017.

[10] A. Filali, A. Kobbane, M. Elmachkour, and S. Cherkaoui, "Sdn controller assignment and load balancing with minimum quota of processing capacity," in *Proceedings of IEEE ICC*, 2018.

[11] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and Y. J. Guo, "Multi-timescale decentralized online orchestration of software-defined networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2716–2730, 2018.

[12] D. T. Nguyen, C. Pham, K. K. Nguyen, and M. Cheriet, "Saco: A service chain aware sdn controller-switch mapping framework," in *Proceedings of CNSM*, 2019.

[13] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: scaling flow management for high-performance networks," in *Proceedings of ACM SIGCOMM*, 2011.

[14] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in *Proceedings of ACM HotSDN*, 2012.

[15] K. Zheng, L. Wang, B. Yang, Y. Sun, Y. Zhang, and S. Uhlig, "Lazyctrl: Scalable network control for cloud data centers," in *Proceedings of IEEE ICDCS*, 2015.

[16] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "Cacheflow: Dependency-aware rule-caching for software-defined networks," in *Proceedings of the Symposium on SDN Research*, 2016.

[17] X. Huang, S. Bian, Z. Shao, and H. Xu, "Dynamic switch-controller association and control devolution for sdn systems," in *Proceedings of IEEE ICC*, 2017.

[18] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[19] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[20] X. Huang, Y. Tang, Z. Shao, H. Xu, and Y. Yang, "Joint switch-controller association and control devolution for sdn systems: An integration of online control and online learning," Technical report, available at: http://faculty.sist.shanghaitech.edu.cn/faculty/shaozy/lasac.pdf.

[21] T. Das, V. Sridharan, and M. Gurusamy, "A survey on controller placement in sdn," *IEEE Communications Surveys & Tutorials*, no. 1, pp. 472–503, 2019.

[22] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 473–478, 2012.

[23] T. Zhang, A. Bianco, and P. Giaccone, "The role of inter-controller traffic in sdn controllers placement," in *Proceedings of IEEE NFV-SDN*, 2016.

[24] M. He, A. Basta, A. Blenk, and W. Kellerer, "Modeling flow setup time for controller placement in sdn: Evaluation for dynamic flows," in *Proceedings of IEEE ICC*, 2017.

[25] M. Tanha, D. Sajjadi, and J. Pan, "Enduring node failures through resilient controller placement for software defined networks," in *Proceedings of IEEE GLOBECOM*, 2016.

[26] B. P. R. Killi and S. V. Rao, "Capacitated next controller placement in software defined networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 514–527, 2017.

[27] ——, "On placement of hypervisors and controllers in virtualized software defined network," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 840–853, 2018.

[28] X. Huang, S. Bian, Z. Shao, and H. Xu, "Predictive switch-controller association and control devolution for sdn systems," in *Proceedings of IEEE/ACM IWQoS*, 2019.

[29] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 393–430, 2019.

[30] S. Sendra, A. Rego, J. Lloret, J. M. Jimenez, and O. Romero, "Including artificial intelligence in a routing protocol using software defined networks," in *Proceedings of IEEE International Workshop on Smart Communication Protocols and Algorithms (SCPA)*, 2017.

[31] S.-C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "Qos-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *Proceedings of IEEE SCC*, 2016.

[32] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, "Machine learning in software defined networks: Data collection and traffic classification," in *Proceedings of IEEE ICNP*, 2016.

[33] P. Wang, S.-C. Lin, and M. Luo, "A framework for qos-aware traffic classification using semi-supervised machine learning in sdns," in *Proceedings of IEEE SCC*, 2016.

[34] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1046–1061, 2017.

[35] S. D'Oro, L. Galluccio, S. Palazzo, and G. Schembra, "A game theoretic approach for distributed resource allocation and orchestration of softwarized networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 3, pp. 721–735, 2017.

[36] M. He, P. Kalmbach, A. Blenk, W. Kellerer, and S. Schmid, "Algorithm-data driven optimization of adaptive communication networks," in *Proceedings of IEEE ICNP*, 2017.

[37] M. H. Rehmani, F. Akhtar, A. Davy, and B. Jennings, "Achieving resilience in sdn-based smart grid: A multi-armed bandit approach," in *Proceedings of IEEE NetSoft*, 2018.

[38] T. Lattimore and C. Szepesvári, "Bandit algorithms," *preprint arXiv: 1510.00757*, 2018.

[39] D. Bouneffouf and I. Rish, "A survey on practical applications of multi-armed and contextual bandits," *preprint arXiv:1904.10040*, 2019.

[40] X. Gao, X. Huang, Y. Tang, Z. Shao, and Y. Yang, "Proactive cache placement with bandit learning in fog-assisted iot systems," in *Proceedings of IEEE ICC*, 2020.

[41] J. Zhu, X. Huang, and Z. Shao, "Learning-aided content placement in caching-enabled fog computing systems using thompson sampling," in *Proceedings of IEEE ICASSP*, 2020.

[42] J. Zhu, X. Huang, X. Gao, Z. Shao, and Y. Yang, "Multi-interface channel allocation in fog computing systems using thompson sampling," in *Proceedings of IEEE ICC*, 2020.

[43] S. Li, A. Karatzoglou, and C. Gentile, "Collaborative filtering bandits," in *Proceedings of ACM SIGIR*, 2016, pp. 539–548.

[44] I. Aykin, B. Akgun, M. Feng, and M. Krunz, "Mamba: A multi-armed bandit framework for beam tracking in millimeter-wave systems," in *Proceedings of IEEE INFOCOM*, 2020.

[45] F. Li, J. Liu, and B. Ji, "Combinatorial sleeping bandits with fairness constraints," in *Proceedings of IEEE INFOCOM*, 2019.

[46] R. Ford, M. Zhang, M. Mezzavilla, S. Dutta, S. Rangan, and M. Zorzi, "Achieving ultra-low latency in 5g millimeter wave cellular networks," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 196–203, 2017.

[47] A. Mantas and F. Ramos, "Rama: Controller fault tolerance in software-defined networking made practical," *preprint arXiv:1902.01669*, 2019.

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

[49] R. Agrawal, "Sample mean based index policies by $o(\log n)$ regret for the multi-armed bandit problem," *Advances in Applied Probability*, vol. 27, no. 4, pp. 1054–1078, 1995.

[50] ONF, "Openflow specification," https://www.opennetworking.org/software-defined-standards/specifications/.

[51] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of ACM SIGCOMM*, 2010.

[52] J.-Y. Audibert and S. Bubeck, "Minimax policies for adversarial and stochastic bandits," in *Proceedings of COLT*, 2009.

[53] O.-A. Maillard, R. Munos, and G. Stoltz, "A finite-time analysis of multi-armed bandits problems with kullback-leibler divergences," in *Proceedings of COLT*, 2011.