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

Xi Huang, *Member, IEEE*, Yinxu Tang, Ziyu Shao*, *Senior Member, IEEE*,
Yang Yang, *Fellow, IEEE*, Hong Xu, *Senior Member, IEEE*

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 time-averaged regret bound over a finite time horizon.

Index Terms—Software-defined networking, switch-controller association, control devolution, learning-aided control.

I. INTRODUCTION

OVER the past decade, software-defined networking (SDN) has emerged to facilitate more efficient network management and more flexible network programmability [2]. 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 [3]

X. Huang is with School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China, also with the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China, and the University of Chinese Academy of Sciences, Beijing 100049, China. (E-mail: huangxi@shanghaitech.edu.cn)

Y. Tang and Z. Shao are with School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China. (E-mail:{tangyx, shaozy}@shanghaitech.edu.cn)

Y. Yang is with Shanghai Institute of Fog Computing Technology (SHIFT), SCA and SIST, ShanghaiTech University, Shanghai 201210, China, and also with the Research Center for Network Communication, Peng Cheng Laboratory, 2 Xingke First Street, Nanshan, Shenzhen 518000, China. (E-mail: yangyang@shanghaitech.edu.cn)

H. Xu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. (E-mail: hongxu@cuhk.edu.hk)

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 [4]–[15]. 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 [16]–[19], 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 the stability of queues on switches and controllers that buffer untreated requests in SDN systems [20]. The other concern comes from various *uncertainties* in SDN systems. For example, in practice, variations of request arrivals can change dramatically across different timescales [21] [22]. Besides, in some scenarios such as wireless SDN systems [23], switches should communicate with controllers via wireless connections. Such dynamics may be highly time-varying and lead to uncertainties in controller accessibility. Statistics (*e.g.*, the distribution of request inter-arrival times) about such uncertainties are usually not given as prior information in practice. Besides, switches' communication costs incurred or computation costs are also not available upon decision making. Instead, they 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 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 [24] and bandit learning [25]. More specifically, our key contributions are summarized as follows.[1]

- ◇ **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 [24] 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* [25], 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 time-averaged regret within a sublinear $O(\sqrt{\log T/T})$ 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. More detailed proofs are given in our technical report [27].

II. RELATED WORK

Optimizations for SDN Systems: So far, numerous existing works [28] have been conducted to optimize the performance (*e.g.*, the control latency [29]–[31], resiliency [32]–[34], cost efficiency [32], *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 [4]–[15] and control devolution [16]–[19], respectively. Instead of studying these two topics

---

[1]As an extension of our previous IWQoS work [26], this paper provides both proof sketches and complete proofs of our performance analysis, as well as further explanations of our problem formulation and more comprehensive simulation results to evaluate the interplay between online control and online learning in SDN systems.

separately, later works [20] [35] 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 their studied problem is similar to our work, we would like to point out that they 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 per-time-slot system dynamics (*e.g.*, queue backlog sizes of controllers but not system costs). Generalizing to such settings brings great challenges and complexities to the scheme design, which are fully addressed in this paper. Besides, our approach does not assume any particular traffic statistics and applies to arbitrary type of request traffic (even those with great traffic variability and burstiness). Moreover, our approach can also be integrated with state-of-the-art traffic burstiness mitigation techniques [21] [22], bandwidth resource allocation approaches [36] [37] and flow scheduling schemes [38] to further improve system performances.

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 [39] such as those for routing optimization [40] [41], traffic classification [42] [43], and resource management [44]–[46]. 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.* [47] 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. Different from such works, our work studies the problem of joint switch-controller association and control devolution with unknown system dynamics in SDN systems. Accordingly, we not only devise an effective scheme with performance guarantee but also characterize the interplay between control and learning. To our best knowledge, it is the first systematic study on the joint control and learning for SDN systems.

Learning-aided Control under Bandit Settings: To date, the multi-armed bandit (MAB) model [48] has been extensively adopted to study sequential decision-making problems under uncertainty within a wide range of scenarios [49], such as the cache placement in wireless caching systems [50] [51], the channel allocation in fog systems [52], collaborative filtering in recommendation systems [53], and beam tracking in millimeter-wave communication systems [54]. Among varieties of MAB settings, the most related to our work is the combinatorial multi-armed bandit model in [55]. 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 devise an effective scheme that solves the problem with a sublinear time-averaged regret bound while characterizing the interplay between online control and online learning. Our investigations provide interesting insights for the study of learning-aided approaches.
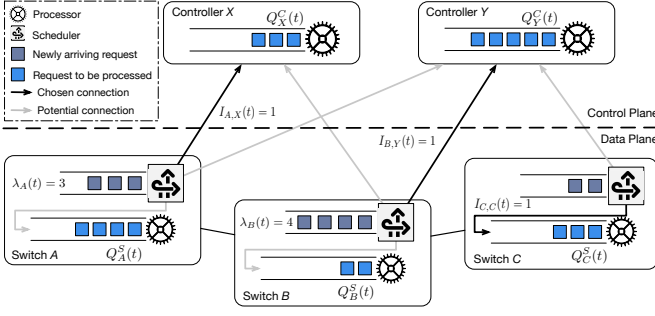
Fig. 1. An instance of our system model in time slot t, with three switches in the data plane and two controllers in the control plane. Each switch i generates a number of $\lambda_i(t)$ new requests and the scheduler makes decision $\{I_{i,j}(t)\}_{j \in \{i,X,Y\}}$ to keep the new requests processed 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 C) and a set of switches (denoted by set $\mathcal{S}$), respectively. Each switch $i \in \mathcal{S}$ has potential associations to a subset $C_i \subseteq C$ 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 $\mathcal{P}(C_i)$ as the power set of C_i, we assume that $p(Z_i) = \Pr\{A_i(t) = Z_i\}$ for each $Z_i \in \mathcal{P}(C_i)$. 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.[2]

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* [2]. 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.

During such a process, owing to recently developed high-speed transmission techniques [56], 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

---

[2]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.

processing queue to buffer them. In the following subsections, 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 $\{I_{i,j}(t)\}_{j \in C_i}$. Specifically, $I_{i,j}(t) = 1$ indicates that controller j is chosen to process new requests and zero otherwise.[3] For simplicity, we use $\mathbf{I}_i(t) \triangleq \{I_{i,i}(t)\} \cup \{I_{i,j}(t)\}_{j \in C}$ 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 [57], each switch i is restricted to choose at most one controller in each time slot. Then we have

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

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

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

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 $j \in C$,

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

E. Optimization Objectives:

On the one hand, the benefit from uploading requests lies in that controllers have adequate service capacities and thus they incur shorter processing latencies than switches. Such a benefit may be offset unexpectedly by considerable communication costs (*e.g.*, round-trip time [20]) 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

---

[3]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.

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

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

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* [58] implies that no requests will ever experience excessively long queueing delay. In this paper, we define queue stability [24] as

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

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 [24]. 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 queue backlogs with empirically lowest costs, then 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) [55]. 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 algorithm design and performance analysis.

A. Motivation for Reformulation

The multi-armed bandit (MAB) model [48] has been extensively adopted to study a wide range of sequential decision-making problems under uncertainty [49]. Under the canonical settings of MAB, a player is considered to play a multi-armed bandit over a finite number of rounds. In each round, the player pulls one of the arms and receives a reward that is sampled from some distribution with an unknown mean for the selected arm. The objective of MAB 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 each time the decision maker (the player) picks only a single choice. 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 is not feasible under 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 their subsequent decision making. Second, unlike the settings of MAB model, the set of arms for the decision maker keeps change across different time slots. Therefore, the basic MAB model does not apply to our problem settings in the face of the coupled nature of decision making among switches and the time-varying availability of switch-controller connections.

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
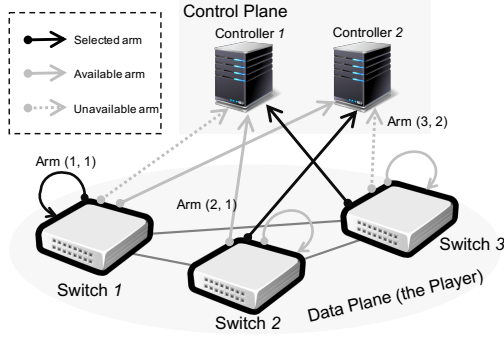
Fig. 2. An illustration of our problem reformulation. During each time slot, the data plane (as the player) has access to a set of connections between switches and controllers. Each accessible connection is viewed as an available arm (denoted by a solid arrow in the figure), while the other connections are unavailable (each denoted by a dashed arrow). The player aims to pick a subset of arms (*e.g.*, arms $(1, 1)$, $(2, 2)$, and $(3, 1)$) so as to maximize its cumulative rewards (*i.e.*, minimize the cumulative system costs) in the long run with queue stability guarantee.

to maximize its cumulative rewards (*i.e.*, equivalent to the minus of its incurred costs) over time slots. We find that such a setting is well defined by the combinatorial multi-armed bandit (CMAB) model with sleeping arms [55]. In the following subsections, we formalize the settings of CMAB and demonstrate our problem reformulation in detail.

B. Combinatorial Bandit Setting

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 time slot t, only a subset of arms are available to the player. We denote their arms by set $A(t) \in \mathcal{P}(\mathcal{N})$. The availability of arms follows a fixed but unknown distribution $p(Z) = \Pr\{A(t) = Z\}$ for $Z \in \mathcal{P}(\mathcal{N})$. Then given $A(t)$, the player needs to choose and pull a subset of arms $f(t)$ (*a.k.a.* *super arm*) whose size is no more than some constant m. Accordingly, we define $\mathcal{F}(A(t)) \triangleq \{f \subseteq A(t) : |f| \leq m\}$ as the set of feasible super arms. For each arm $n \in f(t)$, it is associated with some reward $X_n(t)$ 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 $\mathbb{E}\{\frac{1}{T}\sum_{t=0}^{T-1} R(t)\}$) over T time slots. Such a goal is also equivalent to minimizing reward loss (regret) due to decision making under uncertainty. The regret is characterized by the difference between the maximum achievable reward given the full knowledge of system dynamics and the time-average expected reward under policy π, *i.e.*,

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

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 conduct problem reformulation by fitting our model into the above settings. An illustration of such a reformulation is given in Figure 2.

C. Problem Reformulation

We regard each potential connection between switches and controllers as an arm and construct the arm set $\mathcal{N}$ as $\mathcal{N} \triangleq \{(i,i)\}_{i \in \mathcal{S}} \cup \{(i,j)\}_{i \in \mathcal{S}, j \in C_i}$. 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 $j \in C_i$, 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 $A(t) \triangleq \{(i,i)\}_{i \in \mathcal{S}} \cup (\bigcup_{i \in \mathcal{S}} A_i(t))$. 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 $\mathcal{F}(A(t))$ by

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

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

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

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 [55], 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* [25] to estimate each arm's quality based on its feedback reward information. As an extended version of classic UCB1 (upper-confidence-bound1) method [26], 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 [25].

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 [4]

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

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. In the exploration term, $V_{i,k}$ is defined as

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

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

Regarding online control, we adopt Lyapunov drift techniques [24] 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.

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.[5]

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 maximization and queue stability by jointly considering each arm's reward estimate and its associated instant queue backlog

---

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

[5]In Algorithm 1, the value of $h_{i,k}(-1)$ is set as 0. Besides, in line 8, if more than one candidate achieves the minimum value, then pick one of such optimal candidates uniformly randomly as the target to forward requests.

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 $t \in \{0, 1, \ldots, T-1\}$:
2:     **for** each switch $i \in \mathcal{S}$:
3:         **for** each candidate $k \in \{i\} \cup C_i$:
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

$$l_{i,k}(t) \triangleq \begin{cases} Q_i^S(t) - V \cdot \tilde{x}_{i,i}(t) & \text{if } k = i, \\ Q_k^C(t) - V \cdot \tilde{x}_{i,k}(t) & \text{otherwise.} \end{cases} \tag{17}$$

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).

---

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 $O(|C|)$. 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 manner, which are visualized in Figures 2(a) and 2(b), respectively.

Under centralized implementation, LASAC can be deployed on a particular server that is independent of both the control plane and the data plane. At runtime, LASAC should collect instant system dynamics including the queue backlog sizes on switches and controllers, as well as the communication and computation costs, to conduct the decision making. Then it will spread such decisions onto switches, so that switches can schedule their requests accordingly. 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 [59]. 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
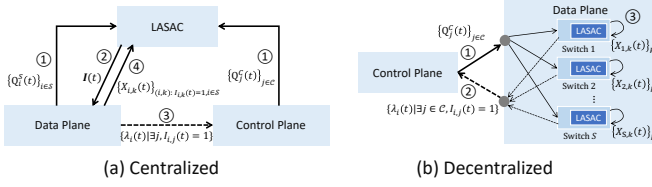
Fig. 3. Two implementations of LASAC

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 a way requires modification on switches, compared to the centralized implementation, it requires less amounts of information exchange and thus 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 $\boldsymbol{\mu} = (\bar{\mu}_1^S, \ldots, \bar{\mu}_{|\mathcal{S}|}^S, \bar{\mu}_1^C, \ldots, \bar{\mu}_{|C|}^C)$, 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

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

Proof Sketch: We leverage Lyapunov drift techniques [24] to conduct the proof. First, we introduce a Lyapunov drift function to characterize the change of queue backlog sizes between successive time slots. Then we switch our focus onto the notion of drift-plus-penalty function, which integrates the objective function into the drift function. Based on the assumption that the total service capacity of the system is greater than the total mean traffic rate, we derive an upper-bound on the drift-plus-penalty function, which is independent of any particular policy. Finally, by exploiting telescoping sum techniques, we obtain the desired result. More details of the proof are given in Appendix A.

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-averaged regret (9) of LASAC has an $O(\sqrt{\log T / T})$ sublinear upper bound; *i.e.*, there exists some positive constant $\tilde{B}$ such that

$$R_{\text{LASAC}}(T) \leq \frac{\tilde{B}}{V} + 2|S| \cdot (|C| + 1) \cdot \left[\beta \sqrt{\frac{\log T}{T}} + \frac{1}{T} (G_\beta + \frac{1}{2}) \cdot \max\{w_{\max}, m_{\max}\} \right], \quad (19)$$

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

Proof Sketch: First, we introduce the notion of per-time-slot regret to measure the difference between the expected rewards that are achieved under the optimal policy and LASAC. Based on such a notion, we derive an upper-bound for the drift-plus-regret term that characterizes the performance difference between the LASAC and an auxiliary policy with a linear function. Then by dividing such a bound into three sub-terms, we adopt techniques such as Chernoff-Hoeffding bound and Jensen's inequality to bound each sub-term to complete the proof. More details of the proof are given in Appendix B.

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 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 10*ms*. 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
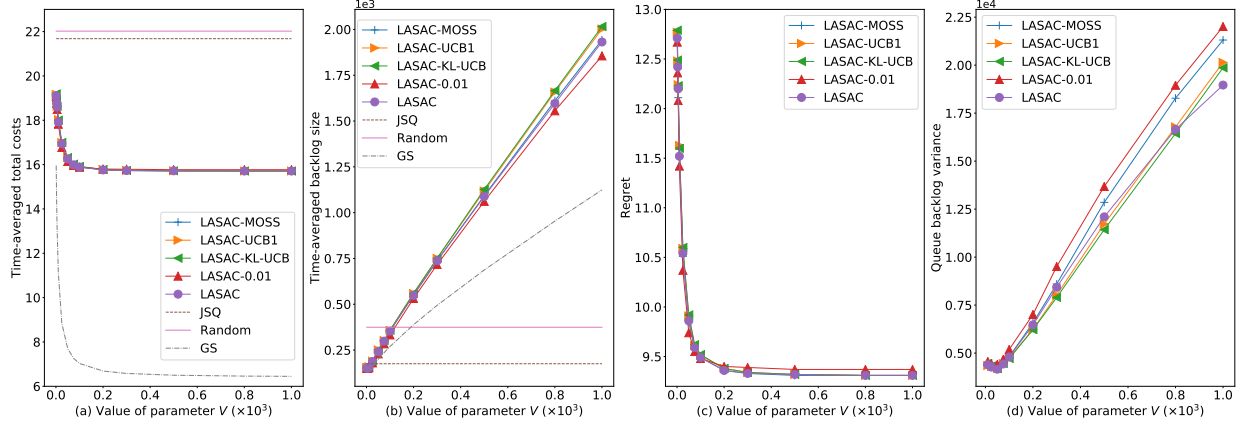
Fig. 4. Performance Comparison under Different Values of V with $\beta = 2$

of real-world networks [60]. 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 run on a per-time-slot basis.

⋄ Greedy Scheduling (GS) [20]: 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 forwards requests to itself or some accessible controller.

⋄ Join-the-shortest-queue (JSQ): Each switch chooses the controller (or itself) with the smallest 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: It follows the same decision making as LASAC, except that switches replace the reward estimate (15) by other UCB variant X, including UCB1 [26], MOSS [61], and KL-UCB [62].

B. Simulation Results and Analysis

We evaluate the performance of LASAC by comparing it with baseline schemes and investigating how it handles the tradeoff between system cost reduction and long-term queue stability, and the interplay between online learning and online control under different parameter settings.

Performance under different choices of parameter V: In Figures 4(a) - 4(d), we set $\beta = 2$ and evaluate different schemes as the value of V increases from 1 to 1000.[6]

Figures 4(a) and 4(b) show that GS achieves near-optimal total costs as the value of V increases to 1000. This is owing

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

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 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 21.6% reduction in system costs and eventually converge thereafter. The corresponding regrets of LASAC and its variants are shown in Figure 4(c). The cost reduction comes with linear growth in the total queue backlog size and, by *Little's theorem* [58], a longer request delay. As shown in Figure 4(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).

Regrets over time under different settings: In Figure 5, we show how the time-averaged regret of LASAC changes over time under different settings of parameters V and β. Particularly, we fix the value of β as 2, 3, 5, and 8 in Figures 5(a) - 5(d), respectively. Meanwhile, the values of parameter V vary from 1 to 100 in each subfigure.

From the results, we first see that in general, when the value of β is fixed, the regret does not grow linearly over time. Instead, the regret increases initially and then gradually diminishes. The reason is that to achieve effective online learning, each switch needs to conduct sufficient explorations in the early stage. In this process, requests may be forwarded over connections with inferior costs, resulting in a high regret. Then such a regret continues to decrease as each switch learns from more and more feedback information to improve its decision making under LASAC. Nonetheless, the regret will not approach zero since the decision-making process should also ensure the stability of queue backlogs in the system. Besides, we also see that given the value of β, the greater
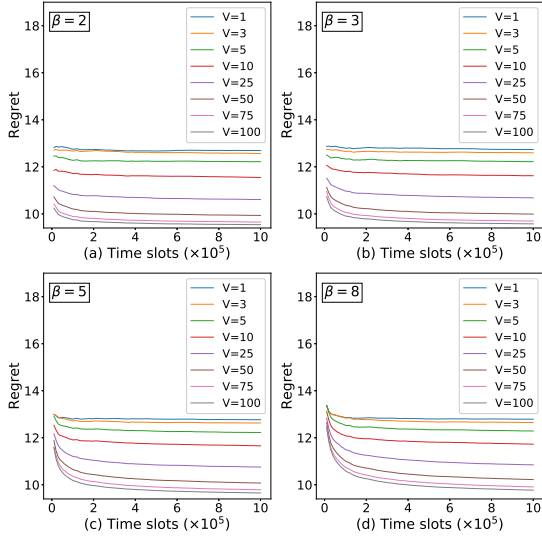
Fig. 5. Regret over Time under Different Settings

Fig. 6. Time-averaged Performance under Different Values of β

the value of parameter V, the lower the regret. This implies that increasing the value of V conduces to regret reduction. Such results verify our results in Theorem 2.

When it comes to the regrets under different settings of parameter β, we see that the larger the value of β, the greater the regret in the early stage. For the reason, recall that the value of β measures the score of exploration in (15). As a result, under LASAC, each switch performs more proactively explorations and incurs a higher regret. Later, such regret curves descend rapidly since more explorations conduce to learning efficiency. The lesson learned is that the value of β should not be set too large; otherwise, the resulting over-explorations would adversely effect system performance, offsetting the benefits of exploration.

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 6(a) - 6(d).

First, we focus on the impact of parameter β on system performance. Figures 6(a) and 6(b) show that larger values of β generally lead to higher total system costs (correspondingly larger regrets, as shown in Figure 6(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 6(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
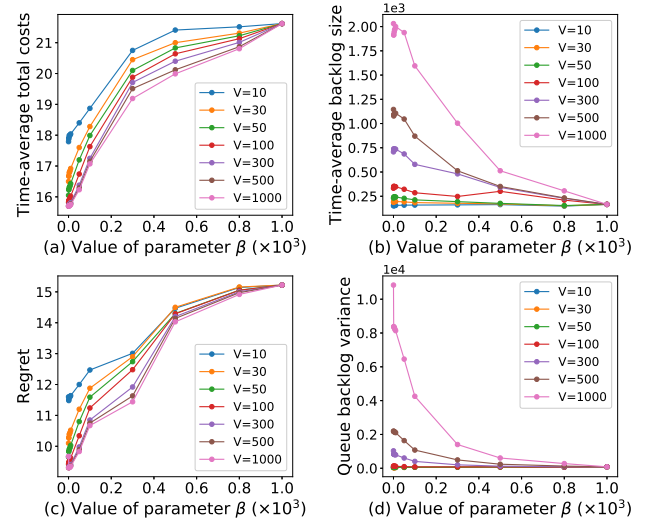
become overloaded due to pure exploitation, LASAC would have the switches turn to other controllers. In other words, exploration is implicitly enforced by the online control procedure to ensure queue stability.

From Figure 4 we know that larger values of V lead to lower system costs and larger total backlog size. Nevertheless, Figure 6 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 sublinear time-averaged 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. Besides, in SDN systems, it is a common practice to retain offline historical information about system operations and performance costs for better network management and troubleshooting. It would be an interesting future work to consider how to integrate such information with online control and online learning to further improve system performance.

REFERENCES

[1] X. Huang, Y. Tang, Z. Shao, Y. Yang, and H. Xu, "Joint switch-controller association and control devolution for sdn systems: An integration of online control and online learning," in *Proceedings of IEEE/ACM IWQoS*, 2020.
[2] 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.
[3] A. Gushchin, A. Walid, and A. Tang, "Scalable routing in sdn-enabled networks with consolidated middleboxes," in *Proceedings of ACM SIGCOMM HotMiddlebox*, 2015.
[4] 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.

[5] 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.

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] M. He, A. Varasteh, and W. Kellerer, "Toward a flexible design of sdn dynamic control plane: An online optimization approach," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1694–1708, 2019.

[15] S. Bera, S. Misra, and N. Saha, "Traffic-aware dynamic controller assignment in sdn," *IEEE Transactions on Communications*, vol. 68, no. 7, pp. 4375–4382, 2020.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] F. Liu, J. Guo, X. Huang, and J. C. Lui, "eba: Efficient bandwidth guarantee under traffic variability in datacenters," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 506–519, 2016.

[22] P. Jin, J. Guo, Y. Xiao, R. Shi, Y. Niu, F. Liu, C. Qian, and Y. Wang, "Postman: Rapidly mitigating bursty traffic by offloading packet processing," in *Proceedings of USENIX ATC*, 2019.

[23] A. Betzler, D. Camps-Mur, E. Garcia-Villegas, I. Demirkol, and J. J. Aleixendri, "Sodalite: Sdn wireless backhauling for dense 4g/5g small cell networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1709–1723, 2019.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

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

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

[31] 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.

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

[33] 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.

[34] ——, "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.

[35] 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.

[36] J. Guo, F. Liu, J. C. Lui, and H. Jin, "Fair network bandwidth allocation in iaas datacenters via a cooperative game approach," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 873–886, 2015.

[37] J. Guo, F. Liu, T. Wang, and J. C. Lui, "Pricing intra-datacenter networks with over-committed bandwidth guarantee," in *Proceedings of USENIX ATC*, 2017.

[38] T. Wang, H. Xu, and F. Liu, "Aemon: Information-agnostic mix-flow scheduling in data center networks," in *Proceedings of APNet*, 2017.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

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

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

[50] 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.

[51] 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.

[52] 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.

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

[54] 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.

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

[56] 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.

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

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

[59] ONF, "OpenFlow specification," available at: "https://www.open networking.org/software-defined-standards/specifications/".

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

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

[62] 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.

APPENDIX A
PROOF OF THEOREM 1

We show that LASAC can achieve queueing stability (7) by applying Lyapunov drift analysis techniques [24].

First, we define $\boldsymbol{Q}(t) \triangleq \left\{Q_i^S(t)\right\}_{i \in S} \cup \left\{Q_j^C(t)\right\}_{j \in C}$ as the set of all queue backlogs in the system during time slot t. Then we consider the following Lyapunov function:

$$L(\boldsymbol{Q}(t)) \triangleq \frac{1}{2}\sum_{i \in S}\left(Q_i^S(t)\right)^2 + \frac{1}{2}\sum_{j \in C}\left(Q_j^C(t)\right)^2. \quad (20)$$

Next, we define the Lyapunov drift Δ_t as the difference between $L(\boldsymbol{Q}(t+1))$ and $L(\boldsymbol{Q}(t))$, and constant B as

$$B \triangleq \frac{1}{2}(|S|+|C|)\cdot \mu_{\max}^2 + \frac{1}{2}|S|\left(\lambda_{\max}\right)^2, \quad (21)$$

then by (17) and the non-positiveness of $\tilde{x}_{i,k}(t)$, we have

$$\begin{aligned}
\Delta_t \leq \ & B - \sum_{i \in S} Q_i^S(t)\mu_i^S(t) - \sum_{j \in C} Q_j^C(t)\mu_j^C(t) \\
& + \sum_{i \in S} \lambda_i(t)\left[I_{i,i}(t)\left(Q_i^S(t) - V \cdot \tilde{x}_{i,i}(t)\right)\right] \\
& + \sum_{i \in S} \lambda_i(t)\left[\sum_{j \in C} I_{i,j}(t)\left(Q_j^C(t) - V \cdot \tilde{x}_{i,j}(t)\right)\right],
\end{aligned} \quad (22)$$

Next we denote the mean vector across all service capacities by $\hat{\boldsymbol{\mu}} \triangleq \left(\mathbb{E}\left[\mu_i^S\right], \ldots, \mathbb{E}\left[\mu_j^C\right]\right)$. Such a rate vector is said to be *feasible* if there exists a policy that makes decisions over time slots to ensure the stability constraint (7). We denote the set of all such feasible vectors by feasible region Γ. Following this definition, we assume that the vector $\hat{\mu}$ is strictly within region Γ, thus there exists a stationary and randomized policy α, characterized by a probability distribution $\boldsymbol{q} = \left[q_f(Z), \forall f \in \mathcal{F}(Z), \forall Z \in \mathcal{P}(\mathcal{N})\right]$, such that $\sum_{f \in \mathcal{F}(Z)} q_f(Z) = 1$ while satisfying the queue stability guarantee (7). Under such a policy, given set of available super arms, $\mathcal{F}(Z)$, policy α picks a super arm f with probability $q_f(Z)$ independently across time slots; then for policy α, there exists some positive constant ϵ such that

$$\sum_{Z \in \mathcal{P}(Z)} p(Z)\left\{\sum_{f \in \mathcal{F}(Z)} q_f^\alpha(Z)\sum_{i:(i,k)\in f} \overline{\lambda}_i\right\} \leq \hat{\mu}_k - \epsilon, \forall k \in S \cup C. \quad (23)$$

Taking condition expectation of (22) given $\boldsymbol{Q}(t)$, we have

$$\begin{aligned}
& \mathbb{E}[L(\boldsymbol{Q}(t+1)) - L(\boldsymbol{Q}(t))|\boldsymbol{Q}(t)] \\
& \leq B - \sum_{i \in S} Q_i^S(t)\cdot \overline{\mu}_i - \sum_{j \in C} Q_j^C(t)\cdot \overline{\mu}_j + \\
& \mathbb{E}\left\{\sum_{i \in S}\lambda_i(t)\left(\sum_{k \in \{i\}\cup C} q_{i,k}^\alpha(A(t))Q_k(t)\right)\bigg|\boldsymbol{Q}(t)\right\} - V|S|\lambda_{\max}x_{\min},
\end{aligned} \quad (24)$$

where we define

$$x_{\min} \triangleq -\max\{w_{\max}, m_{\max}\}. \quad (25)$$

Therein (a) is due to the arg-min operator taken by the algorithm (17) during each time slot t. On the right-hand-side of inequality (a), $q_{i,k}^\alpha(A(t))$ refers to the probability that arm (i,k) is chosen under policy α, i.e. $q_{i,k}^\alpha(A(t)) = \sum_{f \in \mathcal{F}(A(t)):(i,k)\in f} q_f^\alpha(A(t))$. (b) holds because $\tilde{x}_{i,k}(t)$ has lower bound $x_{\min}$. Then (c) is yielded based on the fact that policy α's decision making is independent of the queue backlogs in each time slot. Inequality (d) holds due to the boundedness of $\lambda_i(t)$.

By applying (23) and (24), we have

$$\begin{aligned}
& \mathbb{E}[L(\boldsymbol{Q}(t+1)) - L(\boldsymbol{Q}(t))|\boldsymbol{Q}(t)] \\
& \leq \hat{B} - \epsilon\left(\sum_{i \in S} Q_i^S(t) + \sum_{j \in C} Q_j^C(t)\right),
\end{aligned} \quad (26)$$

where $\hat{B}$ is defined as

$$\begin{aligned}
\hat{B}(V) & \triangleq B - V|S|\lambda_{\max}x_{\min} \\
& = B + V|S|\lambda_{\max}\max\{w_{\max}, m_{\max}\}.
\end{aligned} \quad (27)$$

By summing $\mathbb{E}[L(\boldsymbol{Q}(t+1)) - L(\boldsymbol{Q}(t))|\boldsymbol{Q}(t)]$ over $t = 0, \ldots, T-1$ and taking expectation on both sides, we have

$$\begin{aligned}
& \mathbb{E}[L(\boldsymbol{Q}(T)) - L(\boldsymbol{Q}(0))] \\
& \leq T \cdot \hat{B}(V) - \epsilon\sum_{t=0}^{T-1}\mathbb{E}\left\{\sum_{i \in S} Q_i^S(t) + \sum_{j \in C} Q_j^C(t)\right\}.
\end{aligned} \quad (28)$$

Since $\boldsymbol{Q}(T) \geq 0$ and $\boldsymbol{Q}(0) \geq 0$, then by dividing by T at both sides and taking lim-sup as $T \to \infty$ and with the positiveness of ϵ, we have

$$\begin{aligned}
& \limsup_{T \to \infty} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left\{\sum_{i \in S} Q_i^S(t) + \sum_{j \in C} Q_j^C(t)\right\} \\
& \leq \frac{B}{\epsilon} + V \cdot |S| \cdot \lambda_{\max} \cdot \max\{w_{\max}, m_{\max}\}.
\end{aligned} \quad (29)$$

$\blacksquare$

APPENDIX B
PROOF OF THEOREM 2

Consider an optimal policy α^* and its corresponding probability distributions $\mathbf{q}^* = \left[q_Z^*(f), \forall Z \in \mathcal{P}(\mathcal{N}), \forall f \in \mathcal{F}(Z)\right]$. Let $f^*(t)$ be the super arm set selected by policy α^* in round t. Matrix $\mathbf{I}^*(t) \triangleq (I_{i,j})_{i \in S, j \in \{i\}\cup C}$ is the associated action matrix. Accordingly, we have

$$R^* = \sum_{Z \in \mathcal{P}(N)} p(Z)\sum_{f \in \mathcal{F}(Z)} q_Z^*(f)\sum_{n \in f} \overline{x}_n = \mathbb{E}\left[\sum_{(i,j)\in f^*(t)} \overline{x}_{i,j}\right]. \quad (30)$$

By (30) and (9), we can rewrite the regret of LASAC as

$$\begin{aligned}
R_{\text{LASAC}}(T) & = R^* - \mathbb{E}\left\{\frac{1}{T}\sum_{t=1}^T \sum_{n \in f(t)} X_n(t)\right\} \\
& = \frac{1}{T}\sum_{t=1}^T\left\{R^* - \mathbb{E}\left[\sum_{(i,j)\in f(t)} \overline{x}_{i,j}(t)\right]\right\} \\
& = \frac{1}{T}\sum_{t=1}^T\mathbb{E}\left[\sum_{(i,j)\in f^*(t)} \overline{x}_{i,j}(t) - \sum_{(i,j)\in f(t)} \overline{x}_{i,j}(t)\right].
\end{aligned} \quad (31)$$

Next, we define $\Delta R(t)$ as the difference between the expected rewards which are achieved by the optimal policy α^* and LASAC during time slot t. In particular,

$$\begin{aligned}
\Delta R(t) & \triangleq \sum_{(i,j)\in f^*(t)} \overline{x}_{i,j}(t) - \sum_{(i,j)\in f(t)} \overline{x}_{i,j}(t) \\
& = \sum_{i \in S}\left[I_{i,i}^*(t)\overline{x}_{i,i} + \sum_{j \in C} I_{i,j}^*(t)\overline{x}_{i,j}\right] \\
& \quad - \sum_{i \in S}\left[I_{i,i}(t)\overline{x}_{i,i} + \sum_{j \in C} I_{i,j}(t)\overline{x}_{i,j}\right].
\end{aligned} \quad (32)$$

By adding $\Delta R(t)$ (scaled by $V\lambda_{\max}$) to (22), we have

$$L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t)) + V\lambda_{\max}\Delta R(t)$$
$$\leq B + \lambda_{\max}\sum_{i\in\mathcal{S}}\Big[\sum_{k\in\{i\}\cup C}(Q_k(t) - V\overline{x}_{i,k})(I_{i,k}(t) - I_{i,k}^*(t))\Big]. \tag{33}$$

By defining $Q_k(t) \triangleq Q_i^s(t)$ if $k = i$ and $Q_j^c(t)$ otherwise, then taking the expectation of (33), we obtain

$$\mathbb{E}[L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t)) + V\lambda_{\max}\Delta R(t)] \overset{(a)}{\leq} B +$$
$$\lambda_{\max}\mathbb{E}\Big\{\sum_{i\in\mathcal{S}}\Big[\sum_{k\in\{i\}\cup C}(Q_k(t) - V\overline{x}_{i,k})(I_{i,k}(t) - I_{i,k}^*(t))\Big]\Big\}.$$

Next, we define $Z_1(t)$ as follows:

$$Z_1(t) \triangleq \sum_{i\in\mathcal{S}}\Big[\sum_{k\in\{i\}\cup C}(Q_k(t) - V\overline{x}_{i,k})(I_{i,k}(t) - I_{i,k}^*(t))\Big]. \tag{34}$$

Then summing (34) over $t\in\{0,\dots,T-1\}$, dividing both sides of the inequality by $TV\lambda_{\max}$, we have

$$\frac{1}{TV\lambda_{\max}}\mathbb{E}[L(\mathbf{Q}(T)) - L(\mathbf{Q}(0))] + \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\Delta R(t)]$$
$$\leq \frac{B}{V\lambda_{\max}} + \frac{1}{TV}\sum_{t=0}^{T-1}\mathbb{E}\Big[Z_1(t)\Big]. \tag{35}$$

Since $L(\mathbf{Q}(T)) \geq 0$ and $L(\mathbf{Q}(0)) = 0$, then

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\Delta R(t)] \leq \frac{B}{V\lambda_{max}} + \frac{1}{TV}\sum_{t=0}^{T-1}\mathbb{E}\Big[Z_1(t)\Big]. \tag{36}$$

Next, we consider a policy π', in each time slot t, choose a super arm $f'(t)$ as follows

$$f'(t) \in \arg\min\sum_{(i,k)\in f}(Q_k(t) - V\cdot\overline{x}_{i,k}(t)). \tag{37}$$

Therefore, we have

$$\sum_{(i,k)\in f(t)}(Q_k(t) - V\cdot\tilde{x}_{i,k}(t)) \leq \sum_{(i,k)\in f'(t)}(Q_k(t) - V\cdot\tilde{x}_{i,k}(t)), \tag{38}$$

thus we reach an upper bound on $Z_1(t)$, i.e.,

$$Z_1(t) = \sum_{(i,k)\in f(t)}(Q_k(t) - V\overline{x}_{i,k}) - \sum_{(i,k)\in f^*(t)}(Q_k(t) - V\overline{x}_{i,k})$$
$$\overset{(a)}{\leq} \sum_{(i,k)\in f(t)}(Q_k(t) - V\overline{x}_{i,k}) - \sum_{(i,k)\in f'(t)}(Q_k(t) - V\overline{x}_{i,k})$$
$$+ \sum_{(i,k)\in f'(t)}\Big(Q_k(t) - V\tilde{x}_{i,k}(t)\Big) - \sum_{(i,k)\in f(t)}\Big(Q_k(t) - V\tilde{x}_{i,k}(t)\Big)$$
$$= V\Big[\sum_{(i,k)\in f(t)}\Big(\tilde{x}_{i,k}(t) - \overline{x}_{i,k}\Big) + \sum_{(i,k)\in f'(t)}\Big(\overline{x}_{i,k} - \tilde{x}_{i,k}(t)\Big)\Big], \tag{39}$$

where (a) is due to (38). Next, we define $Z_2(t)$ and $Z_3(t)$ as

$$Z_2(t) \triangleq \sum_{(i,k)\in f(t)}\Big(\tilde{x}_{i,k}(t) - \overline{x}_{i,k}\Big) \tag{40}$$

and

$$Z_3(t) \triangleq \sum_{(i,k)\in f'(t)}\Big(\overline{x}_{i,k} - \tilde{x}_{i,k}(t)\Big). \tag{41}$$

In order to bound $Z_2(t)$, we consider an arbitrary arm

$(i,k)_{i\in\mathcal{S},k\in\{i\}\cup C}$ and in time slot t. Let $t_a^{i,k}$ be the time slot during which arm (i,k) is played for the a-th time. We know that $h_{i,k}(t) \triangleq \sum_{\tau=1}^{t}I_{i,k}(\tau)$ which represents the number of arm (i,k) which has been played by the end of round t. Obviously, we have $I_{i,k}(t_a^{i,k}) = 1, h_{i,k}(t_a^{i,k}) = a$ and $h_{i,k}(t_a^{i,k} - 1) = a - 1$ for all $a\in\{1,2,\dots,h_{i,k}(T-1)\}$. Accordingly, we also have

$$0 \leq t_1^{i,k} < t_2^{i,k} < \cdots < t_{h_{i,k}(T-1)}^{i,k} < T. \tag{42}$$

By defining

$$\Delta X_{i,k}(t) \triangleq \Big\{\tilde{x}_{i,k}(t) - \overline{x}_{i,k} > 0\Big\}, \tag{43}$$

and U^c as the complement event of U, and $\mathbb{1}_{\{.\}}$ as the indicator function. The expectation of $Z_2(t)$ is bounded by

$$\mathbb{E}\Big[Z_2(t)\Big] = \sum_{i\in\mathcal{S}}\sum_{k\in\{i\}\cup C}\mathbb{E}\Big[\Big(\tilde{x}_{i,k}(t) - \overline{x}_{i,k}\Big)I_{i,k}(t)\mathbb{1}_{\Delta X_{i,k}(t)}\Big]$$
$$+ \sum_{i\in\mathcal{S}}\sum_{k\in\{i\}\cup C}\mathbb{E}\Big[\Big(\tilde{x}_{i,k}(t) - \overline{x}_{i,k}\Big)I_{i,k}(t)\mathbb{1}_{\Delta X_{i,k}^c(t)}\Big]$$
$$\overset{(a)}{\leq} \sum_{i\in\mathcal{S}}\sum_{k\in\{i\}\cup C}\mathbb{E}\Big[\underbrace{\Big(\tilde{x}_{i,k}(t) - \overline{x}_{i,k}\Big)I_{i,k}(t)\mathbb{1}_{\Delta X_{i,k}(t)}}_{J_1(t)}\Big],$$

where (a) is due to $\tilde{x}_{i,k}(t) - \overline{x}_{i,k} \leq 0$, provided that event $\Delta X_{i,k}^c(t)$ occurs. Next, by defining $J_1(t) \triangleq \big(\tilde{x}_{i,k}(t) - \overline{x}_{i,k}\big)I_{i,k}(t)\mathbb{1}_{\Delta X_{i,k}(t)}$ and another event

$$F_{i,k}(t) \triangleq \Big\{\hat{x}_{i,k}(t-1) - \overline{x}_{i,k} \leq \beta\sqrt{\frac{\log t}{4h_{i,k}(t-1)}}\Big\}, \tag{44}$$

and summing $J_1(t)$ over $t\in\{0,\dots,T-1\}$, we have

$$\sum_{t=0}^{T-1}J_1(t) \overset{(a)}{=} \sum_{a=1}^{h_{i,k}(T-1)}\Big(\tilde{x}_{i,k}(t_a^{i,k}) - \overline{x}_{i,k}\Big)\mathbb{1}_{\Delta X_{i,k}(t_a^{i,k})}$$
$$\overset{(b)}{\leq} -x_{\min} + \sum_{a=2}^{h_{i,k}(T-1)}\Big(\tilde{x}_{i,k}(t_a^{i,k}) - \overline{x}_{i,k}\Big)\mathbb{1}_{\Delta X_{i,k}(t_a^{i,k})}$$

$$= -x_{\min} + \sum_{a=2}^{h_{i,k}(T-1)}\Big[\tilde{x}_{i,k}(t_a^{i,k}) - \overline{x}_{i,k}\Big]\mathbb{1}_{\Delta X_{i,k}(t_a^{i,k})}\mathbb{1}_{F_{i,k}(t_a^{i,k})}$$
$$+ \sum_{a=2}^{h_{i,k}(T-1)}\Big[\tilde{x}_{i,k}(t_a^{i,k}) - \overline{x}_{i,k}\Big]\mathbb{1}_{\Delta X_{i,k}(t_a^{i,k})}\mathbb{1}_{F_{i,k}^c(t_a^{i,k})}$$
$$\overset{(c)}{\leq} -x_{\min} + \sum_{a=2}^{h_{i,k}(T-1)}\Big[\tilde{x}_{i,k}(t_a^{i,k}) - \overline{x}_{i,k}\Big]\mathbb{1}_{\Delta X_{i,k}(t_a^{i,k})\cap F_{i,k}(t_a^{i,k})}$$
$$+ \sum_{a=2}^{h_{i,k}(T-1)}-x_{\min}\mathbb{1}_{F_{i,k}^c(t_a^{i,k})}, \tag{45}$$

where $x_{\min} \triangleq -\max\{w_{\max}, m_{\max}\}$, equality (a) is due to $d_{i,k}(t_a^{i,k}) = 1$ for all $a\in\{1,2,\dots,h_{i,k}(T-1)\}$ and $d_{i,k}(t) = 0$ for other t; inequality (b) is due to $\tilde{x}_{i,k}(t_1^{i,k}) - \overline{x}_{i,k} \leq -x_{\min}$, and (c) is due to $(\tilde{x}_{i,k}(t_1^{i,k}) - \overline{x}_{i,k})\mathbb{1}_{\Delta X_{i,k}} \leq -x_{\min}$.

Next, we define

$$J_2(t_a^{i,k}) \triangleq \Big(\tilde{x}_{i,k}(t_a^{i,k}) - \overline{x}_{i,k}\Big)\mathbb{1}_{\Delta X_{i,k}(t_a^{i,k})\cap F_{i,k}(t_a^{i,k})} \tag{46}$$

and

$$J_3(t_a^{i,k}) \triangleq -x_{\min} \mathbb{1}_{F_{i,k}^c(t_a^{i,k})}. \qquad (47)$$

We want to give upper bounds for $\sum_{a=2}^{h_{i,k}(T-1)} \mathbb{E}[J_2(t_a^{i,k})]$ and $\sum_{a=2}^{h_{i,k}(T-1)} \mathbb{E}[J_3(t_a^{i,k})]$.

First, to bound $\sum_{a=2}^{h_{i,k}(T-1)} \mathbb{E}[J_2(t_a^{i,k})]$, we consider $t_a^{i,k}$ for all $a \in \{2, \ldots, h_{i,k}(T-1)\}$. Suppose that event $F_{i,k}(t_a^{i,k})$ happens. Then we have

$$\hat{x}_{i,k}(t_a^{i,k} - 1) - \overline{x}_{i,k} \leq \beta \sqrt{\frac{\log t_a^{i,k}}{4h_{i,k}(t_a^{i,k} - 1)}}. \qquad (48)$$

From (14) and (15), we also have

$$\tilde{x}_{i,k}(t_a^{i,k}) \leq \hat{x}_{i,k}(t_a^{i,k} - 1) + \beta \sqrt{\frac{\log t_a^{i,k}}{4h_{i,k}(t_a^{i,k} - 1)}}. \qquad (49)$$

Combining (48) and (49), it turns out

$$\tilde{x}_{i,k}(t_a^{i,k}) - \overline{x}_{i,k} \leq 2\beta \sqrt{\frac{\log t_a^{i,k}}{4h_{i,k}(t_a^{i,k} - 1)}}, \qquad (50)$$

which implies that for all $a \in \{2, \ldots, h_{i,k}(T-1)\}$,

$$J_2(t_a^{i,k}) = (\tilde{x}_{i,k}(t_a^{i,k}) - \overline{x}_{i,k}) \mathbb{1}_{\Delta X_{i,k}(t_a^{i,k}) \cap F_{i,k}(t_a^{i,k})}$$
$$\leq 2\beta \sqrt{\frac{\log t_a^{i,k}}{4h_{i,k}(t_a^{i,k} - 1)}}. \qquad (51)$$

Summing $J_2(t_a^{i,k})$ over $a \in 2, \ldots, h_{i,k}(T-1)$ yields

$$\sum_{a=2}^{h_{i,k}(T-1)} J_2(t_a^{i,k}) \overset{(a)}{\leq} \sum_{a=2}^{h_{i,k}(T-1)} 2\beta \sqrt{\frac{\log t_a^{i,k}}{4h_{i,k}(t_a^{i,k} - 1)}}$$
$$\overset{(b)}{\leq} \beta \sqrt{\log T} \sum_{a=2}^{h_{i,k}(T-1)} \frac{1}{\sqrt{a-1}} \qquad (52)$$
$$\leq \beta \sqrt{\log T} \left(1 + \int_1^{h_{i,k}(T-1)} \frac{1}{\sqrt{x}} dx\right)$$
$$\leq 2\beta \sqrt{h_{i,k}(T-1) \log T},$$

where (a) is by applying (51), (b) is due to $t_a^{i,k} \leq T$ and $h_{i,k}(t_a^{i,k}) = a - 1$ for all $a \in \{2, \ldots, h_{i,k}(T-1)\}$. We have

$$\sum_{a=2}^{h_{i,k}(T-1)} \mathbb{E}[J_2(t_a^{i,k})] \leq 2\beta \sqrt{\log T} \mathbb{E}[\sqrt{h_{i,k}(T-1)}]. \qquad (53)$$

Next, we switch to bounding $\sum_{a=2}^{h_{i,k}(T-1)} \mathbb{E}[J_3(t_a^{i,k})]$. By applying Chernoff-Hoeffding bound, we have

$$\mathbb{E}\left[\mathbb{1}_{\{F_{i,k}^c(t_a^{i,k})\}}\right] = \mathbb{P}\{F_{i,k}^c(t_a^{i,k})\} \leq \frac{1}{t_a^{i,k \frac{\beta^2}{2}}}. \qquad (54)$$

Suppose that $\{t_2^i, \ldots, t_{h_i(T-1)}^i\} \subseteq \{1, 2, \ldots\}$, and that $\frac{\beta^2}{2} > 1$, i.e., $\beta > \sqrt{2}$, we have

$$\sum_{a=2}^{h_{i,k}(T-1)} \mathbb{E}\left[\mathbb{1}_{\{F_{i,k}^c(t_a^{i,k})\}}\right] \leq \sum_{a=2}^{h_{i,k}(T-1)} \frac{1}{t_a^{i,k \frac{\beta^2}{2}}} \leq \sum_{t=1}^{\infty} \frac{1}{t^{\frac{\beta^2}{2}}}. \qquad (55)$$

We define $G_\beta \triangleq \sum_{t=1}^{\infty} t^{-\frac{\beta^2}{2}}$ and we know that when $\frac{\beta^2}{2} > 1$,

G has the property of convergence. Therefore,

$$\sum_{a=2}^{h_{i,k}(T-1)} \mathbb{E}\left[J_3(t_a^{i,k})\right] \leq -x_{\min} G_\beta. \qquad (56)$$

Taking expectation at both sides of (45), then combining (53) and (56), we have

$$\sum_{t=0}^{T-1} \mathbb{E}[J_1(t)] \leq -(G_\beta + 1)x_{\min} + 2\beta \sqrt{\log T} \mathbb{E}[\sqrt{h_{i,k}(T-1)}]. \qquad (57)$$

By summing (44) over $t \in \{0, \ldots, T-1\}$ and plugging (57) thereinto, we have

$$\sum_{t=0}^{T-1} \mathbb{E}[Z_2(t)] \leq \sum_{i \in \mathcal{S}} \sum_{k \in \{i\} \cup \mathcal{C}} -(G_\beta + 1)x_{\min}$$
$$+ \sum_{i \in \mathcal{S}} \sum_{k \in \{i\} \cup \mathcal{C}} \left(2\beta \sqrt{\log T} \mathbb{E}[\sqrt{h_{i,k}(T-1)}]\right) \qquad (58)$$
$$\overset{(a)}{\leq} |\mathcal{S}|(|\mathcal{C}| + 1)\left(2\beta \sqrt{T \log T} - (G_\beta + 1)x_{\min}\right).$$

Next, we switch to bounding $Z_3(t)$. Consider an arbitrary arm $(i,k)_{i \in \mathcal{S}, k \in \{i\} \cup \mathcal{C}}$ and arbitrary time slot $t = 0, 1, \ldots, T-1$. Recall that $Z_3(t) = \sum_{(i,k) \in f'(t)} (\overline{x}_{i,k} - \tilde{x}_{i,k}(t))$. Let matrix $\mathbf{I}'(t) \triangleq \{I'_{i,k}\}_{i \in \mathcal{S}, k \in \{i\} \cup \mathcal{C}}$ be the action matrix corresponding to $f'(t)$. Also, recall that $\Delta X_{i,k}(t) \triangleq \{\tilde{x}_{i,k}(t) - \overline{x}_{i,k} > 0\}$. Similar to the derivation for $Z_2(t)$ in (44), we bound the expectation of $Z_3(t)$ as follows

$$\mathbb{E}[Z_3(t)] = \sum_{(i,k) \in f(t)} (\overline{x}_{i,k} - \tilde{x}_{i,k}(t))$$
$$= \mathbb{E}\left[\sum_{i \in \mathcal{S}} \sum_{k \in \{i\} \cup \mathcal{C}} (\overline{x}_{i,k} - \tilde{x}_{i,k}(t)) I'_{i,k}(t)\right] \qquad (59)$$
$$\overset{(a)}{\leq} \sum_{i \in \mathcal{S}} \sum_{k \in \{i\} \cup \mathcal{C}} \mathbb{E}\left[(\overline{x}_{i,k} - \tilde{x}_{i,k}(t)) I_{i,k}(t) \mathbb{1}_{\Delta X_{i,k}^c(t)}\right],$$

where (a) is due to $\overline{x}_{i,k} - \tilde{x}_{i,k}(t) \leq 0$ when the event occurs.

Next, we define $K_1(t) \triangleq (\overline{x}_{i,k} - \tilde{x}_{i,k}(t)) I_{i,k}(t) \mathbb{1}_{\Delta X_{i,k}^c(t)}$. Then we consider the following two cases.

In the first case, i.e., when $t \leq t_1^{i,k}$, event $\Delta X_{i,k}^c(t)$ must not occur. This is because $\overline{x}_{i,k} = 0$ due to $h_{i,k} = 0$ for $t \leq t_1^{i,k}$. Hence, for all $t \leq t_1^{i,k}$, we have $\mathbb{E}[K_1(t)] = 0$.

In the second case, when $t > t_1^{i,k}$, we suppose that event $\Delta X_{i,k}^c(t)$ occurs. Then we have $\overline{x}_{i,k} - \tilde{x}_{i,k}(t) \geq 0$ which, along with (14), implies that

$$\Delta X_{i,k}(t) = \left\{\hat{x}_{i,k}(t-1) - \overline{x}_{i,k} > -\beta \sqrt{\frac{\log t}{4h_{i,k}(t-1)}}\right\}. \qquad (60)$$

Therefore, for all $t > t_1^{i,k}$, we have

$$\mathbb{E}\left[K_1(t)\right] = \mathbb{E}\left[(\overline{x}_{i,k} - \tilde{x}_{i,k}(t))I'_{i,k}(t)\mathbb{1}_{\Delta X^c_{i,k}(t)}\right]$$

$$\overset{(a)}{\leq} -x_{\min}\mathbb{E}\left[\mathbb{1}_{\Delta X^c_{i,k}(t)}\right] \overset{(b)}{\leq} \frac{-x_{\min}}{t^{\frac{\beta^2}{2}}}, \tag{61}$$

where inquality (a) is due to $\overline{x}_{i,k} - \tilde{x}_{i,k}(t) \leq -x_{\min}$ and inequality (b) is derived from Chernoff-Hoeffding bound.

By summing $\mathbb{E}\left[K_1(t)\right]$ over $t \in \{0, \ldots, T-1\}$, we have

$$\sum_{t=0}^{T-1}\mathbb{E}\left[K_1(t)\right] \leq -x_{\min}\sum_{t=t_1^{i,k}+1}^{T-1}\frac{1}{t^{\frac{\beta^2}{2}}}$$

$$\leq -x_{\min}\sum_{t=1}^{\infty}\frac{1}{t^{\frac{\beta^2}{2}}} = -G_\beta x_{\min}. \tag{62}$$

Likewise, we have

$$\sum_{t=0}^{T-1}\mathbb{E}\left[Z_3(t)\right] \leq -|S|(|C|+1)G_\beta x_{\min}. \tag{63}$$

By (36), (25), and $Z_1(t) \leq V(Z_2(t) + Z_3(t))$, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\Delta R(t)] \leq \frac{B}{V\lambda_{\max}} + \frac{1}{TV}\sum_{t=0}^{T-1}\mathbb{E}\left[Z_1(t)\right]$$

$$= \frac{B}{V\lambda_{\max}} + \frac{1}{TV}V\left(\sum_{t=0}^{T-1}\mathbb{E}\left[Z_2(t)\right] + \sum_{t=0}^{T-1}\mathbb{E}\left[Z_3(t)\right]\right)$$

$$\leq \frac{B}{V\lambda_{\max}} + \frac{1}{T}\left[(|S|(|C|+1)\left(2\beta\sqrt{T\log T} - (G_\beta+1)x_{\min}\right)\right] - \frac{1}{T}|S|(|C|+1)G_\beta x_{\min}$$

$$= \frac{B}{V\lambda_{\max}} + \frac{|S|(|C|+1)}{T}\left(2\beta\sqrt{T\log T} - (2G_\beta+1)x_{\min}\right)$$

$$= \frac{B}{V\lambda_{\max}} + \frac{|S|(|C|+1)}{T}[2\beta\sqrt{T\log T} + (2G_\beta+1)\max\{w_{\max}, m_{\max}\}],$$

where constant B is defined in (21) and $G_\beta \triangleq \sum_{t=1}^{\infty}t^{-\frac{\beta^2}{2}}$.

∎

Ziyu Shao is an assistant professor at the School of Information Science and Technology, ShanghaiTech University, Shanghai, China. His current research interests center on Intelligent Networks, including AI for Networks and Networks for AI. He received his B.S. degree and M.Eng. degree from Peking University in 2001 and 2004, respectively, and Ph.D. degree from the Chinese University of Hong Kong in 2010. He then worked as a post-doctoral researcher at the Chinese University of Hong Kong from 2011 to 2013. He was a visiting post-doctoral researcher at the EE department of Princeton University in 2012. He was also a visiting professor at the EECS department of UC Berkeley in 2017.

Yang Yang (Fellow, IEEE) received the B.S. and M.S. degrees in Radio Engineering from Southeast University, Nanjing, China, in 1996 and 1999, respectively, and the PhD degree in Information Engineering from the Chinese University of Hong Kong in 2002. He is currently a full professor with School of Information Science and Technology, as well as the Director of Shanghai Institute of Fog Computing Technology (SHIFT), ShanghaiTech University, China. He is also an Adjunct Professor with the Research Center for Network Communication, Peng Cheng Laboratory, China. Before joining Shang-haiTech University, he has held faculty positions at the Chinese University of Hong Kong, Brunel University, U.K., University College London (UCL), U.K., and SIMIT, CAS, China. Yang's research interests include fog computing networks, service-oriented collaborative intelligence, wireless sensor networks, IoT applications, and advanced testbeds and experiments. He has published more than 260 papers and filed more than 80 technical patents in these research areas. He has been the Chair of the Steering Committee of Asia-Pacific Conference on Communications (APCC) since January 2019. In addition, he is a General Co-Chair of the IEEE DSP 2018 conference and a TPC Vice-Chair of the IEEE ICC 2019 conference.
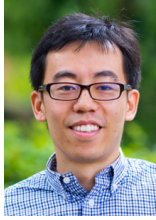
Hong Xu is an Associate Professor in Department of Computer Science and Engineering, The Chinese University of Hong Kong. His research area is computer networking and systems, particularly big data systems and data center networks. From 2013 to 2020 he was with City University of Hong Kong. He received his B.Eng. from The Chinese University of Hong Kong in 2007, and his M.A.Sc. and Ph.D. from University of Toronto in 2009 and 2013, respectively. He was the recipient of an Early Career Scheme Grant from the Hong Kong Research Grants Council in 2014. He received three best paper awards, including the IEEE ICNP 2015 best paper award. He is a senior member of IEEE and member of ACM.

Xi Huang received the B.Eng. degree from Nanjing University, China, in 2014. Since September 2014, he has been with the School of Information Science and Technology at ShanghaiTech University, where he is currently a PhD candidate. He was a visiting student at the Department of Electrical Engineering and Computer Sciences at UC Berkeley, during February to July in 2017. His current research interests include the optimization and the design for intelligent networks.

Yinxu Tang received the B.Eng degree from Shang-haiTech University, Shanghai, China. She is currently pursuing the Master degree at the School of Information Science and Technology, Shang-haiTech University, Shanghai, China. Her current research interests include: reinforcement learning, online control, resource management, and edge intelligence.