The Power of Online Learning in Stochastic Network Optimization

Longbo Huang*, Xin Liu†, Xiaohong Hao*
*{longbohuang haoxh10}@tsinghua.edu.cn, IIIS, Tsinghua University
†liuxin@microsoft.com, Microsoft Research Asia

ABSTRACT

In this paper, we investigate the power of online learning in stochastic network optimization with unknown system statistics *a priori*. We are interested in understanding how information and learning can be efficiently incorporated into system control techniques, and what are the fundamental benefits of doing so. We propose two *Online Learning-Aided Control* techniques, `OLAC` and `OLAC2`, that explicitly utilize the past system information in current system control via a learning procedure called *dual learning*. We prove strong performance guarantees of the proposed algorithms: `OLAC` and `OLAC2` achieve the near-optimal $[O(\epsilon), O([\log(1/\epsilon)]^2)]$ utility-delay tradeoff and `OLAC2` possesses an $O(\epsilon^{-2/3})$ convergence time. Simulation results also confirm the superior performance of the proposed algorithms in practice. To the best of our knowledge, `OLAC` and `OLAC2` are the first algorithms that simultaneously possess explicit near-optimal delay guarantee and sub-linear convergence time, and our attempt is the first to explicitly incorporate online learning into stochastic network optimization and to demonstrate its power in both theory and practice.

1. INTRODUCTION

Consider the following constrained network optimization problem: We are given a stochastic networked system with dynamic system states that have a stationary state distribution. At each state, an operation is implemented and a corresponding system cost occurs depending on the chosen actions. The objective is to minimize the expected cost given service/demand constraints. Such a constrained optimization framework in stochastic systems is general and models many practical application scenarios, such as in computer networks, smart grids, supply chain management, and transportation networks. Due to this wide applicability, developing efficient control techniques for this framework in stochastic systems has been one central question in network optimization.

SIGMETRICS'14, June 16–20, 2014, Austin, Texas, USA.

http://dx.doi.org/10.1145/2591971.2591984.

However, solving this problem is very challenging and the main difficulty comes from the fact that the state distribution of the system is often unknown *a priori*. Moreover, known algorithms that handle this challenge either do not admit explicit delay guarantee, or suffer from a slow convergence speed. To address such a challenge and to overcome the previous limitations, in this paper, we investigate the value of online learning in optimal stochastic system control. We are interested in understanding how information and learning can be efficiently incorporated into system control techniques, and what are the fundamental benefits of doing so.

Specifically, we propose two *Online Learning-Aided Control* techniques, `OLAC` and `OLAC2`. The two new techniques are inspired by the following key aspect of the recently developed Lyapunov technique (also known as Backpressure [1]): the Lyapunov technique converts the problem of finding optimal system control decisions into learning the optimal Lagrange multiplier of an underlying optimization problem in an incremental manner, i.e., at every time, the technique reacts only to the instantaneous system condition. This conversion allows the queues in the system to play the role of Lagrange multipliers, and the incremental nature eliminates the need for knowing the statistical information of the system. Because of this attractive feature, the Lyapunov technique has received much attention in the literature. However, the main limitation of the technique is that under Lyapunov algorithms, the queue size in the system has to build up gradually, which results in a large system delay. Specifically, in order to achieve an $O(\epsilon)$ close-to-optimal performance, the queue size has to be $\Theta(1/\epsilon)$, which is undesirable when ϵ is small.

In comparison, `OLAC` and `OLAC2` explicitly utilize the system information via a learning procedure called *dual learning*, which effectively integrates the past system information into current system control by solving an *empirical* optimal Lagrange multiplier. By doing so, `OLAC` and `OLAC2` convert the problem of optimal control into a combination of stochastic approximation and statistical learning. Note that this is a challenging task, because dual learning introduces another coupling effect in time to stochastic system algorithm performance analysis, which itself is already highly non-trivial due to the complicated interactions among different system components.

To the best of our knowledge, this is the first attempt to explicitly incorporate the power of online learning in stochastic network optimization. We address a general model of stochastic network optimization with unknown system statis-

tics *a priori*. Specifically, we make the following contributions in this paper:

- We propose two Online-Learning-Aided Control algorithms, `OLAC` and `OLAC2`, which explicitly utilize the system information via online learning to integrate the past system information into current system control. `OLAC` and `OLAC2` apply to all scenarios where Backpressure applies. Moreover, they retain all the attractive features of Backpressure, such as having low computational complexities, and assuming zero a priori statistical information. Therefore, they can be applied to large-scale dynamic network systems. Simulation results also demonstrate the empirical effectiveness of our proposed algorithms.
- We prove strong performance guarantees for the proposed algorithms, including near-optimality of the control policies, as well as sub-linear convergence time. Specifically, we show that `OLAC` and `OLAC2` achieve the near-optimal $[O(\epsilon), O([\log(1/\epsilon)]^2)]$ utility-delay tradeoff and `OLAC2` possesses a convergence time of $O(\epsilon^{-2/3})$, which significantly improves upon the $\Theta(1/\epsilon)$ convergence time of the Lyapunov technique. To the best of our knowledge, this is the first result that simultaneously achieves near-optimal utility-delay tradeoffs and sub-linear convergence time, and demonstrates the power of online learning in stochastic network optimization.
- We develop two analytical techniques, *dual learning* and the *augmented problem*, for algorithm design and performance analysis. Dual learning allows us to connect dual subgradient update convergence to statistical convergence of random system states, whereas the augmented problem enables the interplay between Lyapunov drift analysis and duality in analyzing systems that apply time-inhomogeneous control policies. These techniques may be applicable to solving other network control problems.

The rest of the paper is organized as follows. In Section 2, we discuss a few representative examples of stochastic network optimization in diverse application fields and the related works. We then explain how our results advance the state-of-the-art. We set up our notations in Section 3. We present the system model and problem formulation in Section 4, and background information in Section 5. We present `OLAC` and `OLAC2` in Section 6, and prove their optimality in Section 7, and convergence in Section 8. Simulation results are presented in Section 9, followed by conclusions in Section 10.

2. MOTIVATING EXAMPLES

In the following, we present a few representative examples of stochastic network optimization in diverse application fields and the related works, and explain how our main results can be applied.

Wireless Networks Consider the following simplified scheduling problem in cellular networks. A cellular base station (BS) is transmitting data to a mobile user. The channel (i.e., state) between the user and the BS is time varying, and the cost (e.g., energy) and the transmission rate depend on the channel state. The user has a certain arrival rate, and thus the constraint is that the service rate provided by the BS to the user has to be larger than the arrival rate. The objective is to minimize the expected energy consumption, where the expectation is taken over channel distributions, subject to the rate constraint. This example can be generalized in practice, e.g., to include multiple users, multiple hops, multiple transmission rates, various coding/modulation schemes, multiple constraints, and different objectives, e.g., [2], [3].

Smart Grids Consider the following demand response problem with renewable energy sources in a smart grid. The problem is to allocate renewable energy sources (e.g., solar or wind) to flexible consumers (e.g., an EV to be charged or a dishwasher load to be finished). In this case, the renewable energy source provides energy according to a time-varying supply process (state). When the renewable energy source cannot generate enough energy to serve all customer load, it incurs a cost to draw energy from the regular power grid, which has a time-varying price (state). The objective is to minimize the time average cost of using the regular grid (and hence results in the most efficient utilization of the renewable source). The constraint is that the average service rate has to be larger than the arrival rate of the consumer demand. Various related issues can also be considered here, including demand response [4], deferrable load scheduling [5], and energy storage management [6].

Supply Chain Management Consider the following inventory control problem in supply chain management. A manufacturing plant purchases raw materials to assemble products and then sells the final products to customers. There are K types of raw materials, each with a time-varying price (state) and N types of final products, each with a time varying demand (state). At each time instance, the plant needs to decide whether to re-stock each type of raw materials and how to price each final product based on the current and future customer demands and material price. The objective is to maximize profit. In the case of large K and N, approximate dynamic programming solutions [7] and efficient Lyapunov optimization solutions have been proposed [8]. Other issues are studied in general processing networks [9], [10], with applications to semiconductor wafer fabrication facilities and assembly line control.

Transportation Networks Consider the traffic signal control problem in a transportation network. The operator controls each set of traffic signals at traffic intersections to regulate the traffic flow rate in the network. Vehicles enter the network in a random fashion (state), with a constant average rate. They move in the network following pre-specified average turn ratios. The objective is to stabilize the queue and allow vehicles to move as fast as possible. Such a system can be modeled as a "store and forward" queuing network. Feedback policies based on queue measurements have been extensively studied [11], as well as Backpressure based decentralized schemes [12].

Solutions The above examples demonstrate the wide application scenarios of the stochastic optimization problem we consider in this paper. If the distribution of the system state is known or if the system is static, many algorithms have been proposed using flow-based optimization techniques, e.g., [13], [14], and the references therein. However, flow-based schemes typically do not explicitly characterize network delay and algorithm convergence time.

For general stochastic settings, Backpressure algorithms address the challenge of unknown channel state distribu-

tions by gradually building up the queues and use them for optimal decision making, e.g., [15], [3]. However, Backpressure is known for its slow convergence and long delay, because the queues have to be large enough for achieving near-optimal performance. Specifically, for achieving an $O(\epsilon)$ near-optimality, an $\Theta(1/\epsilon)$ queue size is required. There have been recent works trying to obtain improved utility-delay tradeoff for stochastic systems. For instance, [16], [17], and [18] propose algorithms that can achieve the $[O(\epsilon), O([\log(1/\epsilon)]^2)]$ tradeoff. However, all the above algorithms require a convergence time of $\Theta(1/\epsilon)$. Moreover, they typically require additional system knowledge, e.g., system slack, for algorithm design, which adds to the complexity of algorithm implementation.

Our learning-based approach overcomes these limitations. In particular, we use an online learning-based approach to take advantage of the historic state information, which is ignored by Backpressure throughout the system control process. Our approach is based on a novel *dual learning* idea, which computes an empirical Lagrange multiplier based on the empirical distribution of the system states. Then, we include the learned Lagrange multiplier to the network queues for decision making. Two advantages manifest themselves in this learning-based approach. First, using the distribution information gradually learned in the system, we can significantly speed up the convergence to the optimal solution. Second, because of the "virtual" value added to the queue, i.e., the empirical Lagrange multiplier, the actual queue size is significantly smaller than that under Backpressure. Specifically, we develop two Online Learning-Aided Control techniques, `OLAC` and `OLAC2`. We show that `OLAC` and `OLAC2` achieve the $[O(\epsilon), O([\log(1/\epsilon)]^2)]$ tradeoff and `OLAC2` possesses a convergence time of $O(\epsilon^{-2/3})$, which significantly outperforms that of Backpressure.

3. NOTATIONS

$\mathbb{R}^n$ denotes the n-dimensional Euclidean space. $\mathbb{R}^n_+$ and $\mathbb{R}^n_-$ denote the non-negative and non-positive orthant. Bold symbols $\boldsymbol{x} = (x_1, ..., x_n)$ denote vectors in $\mathbb{R}^n$. The notion $w.p.1$ denotes "with probability 1." $\|\cdot\|$ denotes the Euclidean norm. For a sequence of variables $\{y(t)\}_{t=0}^{\infty}$, we also use $\overline{y} = \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{y(\tau)\}$ to denote its average (when exists). $\boldsymbol{x} \succeq \boldsymbol{y}$ means that $x_j \geq y_j$ for all j.

4. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we specify the general network model. We consider a network controller that operates a network with the goal of minimizing the time average cost, subject to the queue stability constraint. The network is assumed to operate in slotted time, i.e., $t \in \{0, 1, 2, ...\}$. We assume there are $r \geq 1$ queues in the network (e.g., the amount of data to be transmitted in cellular networks or the amount of flexible jobs to be scheduled in a smart grid).

4.1 Network State

In every slot t, we use $S(t)$ to denote the current network state, which indicates the current network parameters, such as a vector of conditions for each network link, or a collection of other relevant information about the current network channels and arrivals. We assume that $S(t)$ is i.i.d. over time and takes M different random network states denoted as $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$. [1] Let $\pi_{s_i} = \Pr\{S(t) = s_i\}$ denote the probability of being in state s_i at time t and denote $\boldsymbol{\pi} = (\pi_{s_1}, ..., \pi_{s_M})$ the stationary distribution. We assume that the network controller can observe $S(t)$ at the beginning of every slot t, but the π_{s_i} probabilities are unknown.

4.2 The Cost, Traffic, and Service

At each time t, after observing $S(t) = s_i$, the controller chooses an action $x(t)$ from a set $\mathcal{X}^{(s_i)}$, i.e., $x(t) = x^{(s_i)}$ for some $x^{(s_i)} \in \mathcal{X}^{(s_i)}$. The set $\mathcal{X}^{(s_i)}$ is called the feasible action set for network state s_i and is assumed to be time-invariant and compact for all $s_i \in \mathcal{S}$. The cost, traffic, and service generated by the chosen action $x(t) = x^{(s_i)}$ are as follows:

(a) The chosen action has an associated cost given by the cost function $f(t) = f(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \mapsto \mathbb{R}_+$ (or $\mathcal{X}^{(s_i)} \mapsto \mathbb{R}_-$ in reward maximization problems);

(b) The amount of traffic generated by the action to queue j is determined by the traffic function $A_j(t) = A_j(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \mapsto \mathbb{R}_+$, in units of packets;

(c) The amount of service allocated to queue j is given by the rate function $\mu_j(t) = \mu_j(s_i, x^{(s_i)}) : \mathcal{X}^{(s_i)} \mapsto \mathbb{R}_+$, in units of packets.

Note that $A_j(t)$ includes both the exogenous arrivals from outside the network to queue j, and the endogenous arrivals from other queues, i.e., the transmitted packets from other queues, to queue j. We assume the functions $f(s_i, \cdot)$, $\mu_j(s_i, \cdot)$ and $A_j(s_i, \cdot)$ are time-invariant, their magnitudes are uniformly upper bounded by some constant $\delta_{\max} \in (0, \infty)$ for all s_i, j, and they are known to the network operator.

4.3 Problem Formulation

Let $\boldsymbol{q}(t) = (q_1(t), ..., q_r(t))^T \in \mathbb{R}^r_+$, $t = 0, 1, 2, ...$ be the queue backlog vector process of the network, in units of packets. We assume the following queueing dynamics:

$$q_j(t+1) = \max\big[q_j(t) - \mu_j(t) + A_j(t), 0\big], \quad \forall j, \tag{1}$$

and $\boldsymbol{q}(0) = \boldsymbol{0}$. By using (1), we assume that when a queue does not have enough packets to send, null packets are transmitted, so that the number of packets entering $q_j(t)$ is equal to $A_j(t)$. In this paper, we adopt the following notion of queue stability [1]:

$$\overline{q}_{\text{av}} \triangleq \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{j=1}^{r} \mathbb{E}\{q_j(\tau)\} < \infty. \tag{2}$$

We use Π to denote an action-choosing policy. Then, we use f^{Π}_{av} to denote the time average cost induced by Π, i.e.,

$$f^{\Pi}_{\text{av}} \triangleq \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{f^{\Pi}(\tau)\}, \tag{3}$$

where $f^{\Pi}(\tau)$ is the cost incurred at time τ by policy Π. We call an action-choosing policy *feasible* if at every time slot t it only chooses actions from the feasible action set $\mathcal{X}^{(S(t))}$. We then call a feasible action-choosing policy under which (2) holds a *stable* policy, and use f^*_{av} to denote the optimal time average cost over all stable policies.

In every slot, the network controller observes the current network state and chooses a control action, with the goal of

[1] The results in this paper can likely be generalized to systems with more general Markovian dynamics.

minimizing the time average cost subject to network stability. This goal can be mathematically stated as:

$$\textbf{(P1)}\ \ \textbf{min :}\ \ f_{\text{av}}^{\boldsymbol{\Pi}},\ \text{s.t.}\ \textbf{(2)}.$$

In the following, we call **(P1)** *the stochastic problem*. It can be seen that the examples in Section 2 can all be modeled with the stochastic problem framework. This is the problem formulation we focus on in this paper.

5. THE DETERMINISTIC PROBLEM AND BACKPRESSURE

In this section, we first define the *deterministic problem* and its dual problem, which will be important for designing our new control techniques and for our later analysis. We then review the Lyapunov technique for solving the stochastic problem **(P1)**. To follow the convention, we will call it the Backpressure algorithm.

5.1 The deterministic problem

The *deterministic problem* is defined as follows [17]:

$$\min: F(\boldsymbol{x},\boldsymbol{\pi}) \triangleq V\sum_{s_i}\pi_{s_i}f(s_i,x^{(s_i)}) \tag{4}$$

$$\text{s.t. } H_j(\boldsymbol{x},\boldsymbol{\pi}) \tag{5}$$
$$\triangleq \sum_{s_i}\pi_{s_i}[A_j(s_i,x^{(s_i)})-\mu_j(s_i,x^{(s_i)})]\leq 0,\ \forall j,$$
$$x^{(s_i)}\in\mathcal{X}^{(s_i)}\quad \forall\, i=1,2,...,M.$$

Here the minimization is taken over $\boldsymbol{x}\in\prod_i\mathcal{X}^{(s_i)}$, where $\boldsymbol{x}=(x^{(s_1)},...,x^{(s_M)})^T$, and $V\geq 1$ is a positive constant introduced for later analysis. The dual problem of (4) can be obtained as follows:

$$\max:\ \ g(\boldsymbol{\gamma}),\quad \text{s.t. }\boldsymbol{\gamma}\succeq\mathbf{0}, \tag{6}$$

where $g(\boldsymbol{\gamma})$ is the dual function and is defined as:

$$g(\boldsymbol{\gamma})=\inf_{x^{(s_i)}\in\mathcal{X}^{(s_i)}}\sum_{s_i}\pi_{s_i}\Big\{Vf(s_i,x^{(s_i)}) \tag{7}$$
$$+\sum_j\gamma_j\big[A_j(s_i,x^{(s_i)})-\mu_j(s_i,x^{(s_i)})\big]\Big\}.$$

Here $\boldsymbol{\gamma}=(\gamma_1,...,\gamma_r)^T$ is the *Lagrange multiplier* of (4). It is well known that $g(\boldsymbol{\gamma})$ in (7) is concave in the vector $\boldsymbol{\gamma}$ for all $\boldsymbol{\gamma}\in\mathbb{R}^r$, and hence the problem (6) can usually be solved efficiently, particularly when the cost functions and rate functions are separable over different network components [19].

Below, we use $\boldsymbol{\gamma}^*=(\gamma_1^*,\gamma_2^*,...,\gamma_r^*)^T$ to denote an optimal solution of the problem (6). For notational convenience, we also use $g_0(\boldsymbol{\gamma})$ and $\boldsymbol{\gamma}_0^*$ to denote the dual function and an optimal dual solution for $V=1$. It can be seen that:

$$g(\boldsymbol{\gamma})=Vg_0(\boldsymbol{\gamma}/V), \tag{8}$$

which implies that $\boldsymbol{\gamma}=V\boldsymbol{\gamma}_0^*$ is an optimal solution of $g(\boldsymbol{\gamma})$. Note that $g_0(\boldsymbol{\gamma})$ is independent of V. For our later analysis, we also define:

$$g_{s_i}(\boldsymbol{\gamma})=\inf_{x^{(s_i)}\in\mathcal{X}^{(s_i)}}\Big\{Vf(s_i,x^{(s_i)}) \tag{9}$$
$$+\sum_j\gamma_j\big[A_j(s_i,x^{(s_i)})-\mu_j(s_i,x^{(s_i)})\big]\Big\},$$

to be the dual function when there is only a single state s_i. It is clear from equations (7) and (9) that:

$$g(\boldsymbol{\gamma})=\sum_{s_i}\pi_{s_i}g_{s_i}(\boldsymbol{\gamma}). \tag{10}$$

5.2 The Backpressure algorithm

Among the many techniques developed for solving the stochastic problem, the Backpressure algorithm has received much attention because (i) it does not require any statistical information of the changing network conditions, (ii) it has low implementation complexity, and (iii) it has provable strong performance guarantees. The Backpressure algorithm works as follows [1]. [2]

Backpressure: At every time slot t, observe the current network state $S(t)$ and the backlog $\boldsymbol{q}(t)$. If $S(t)=s_i$, choose $x^{(s_i)}\in\mathcal{X}^{(s_i)}$ that solves the following:

$$\max:\quad -Vf(s_i,x)+\sum_{j=1}^r q_j(t)\big[\mu_j(s_i,x)-A_j(s_i,x)\big] \tag{11}$$
$$\text{s.t.}\quad x\in\mathcal{X}^{(s_i)}.\qquad\diamond$$

In many problems, (11) can usually be decomposed into separate parts that are easier to solve, e.g., [15], [3]. Also, when the network state process $S(t)$ is i.i.d., it has been shown in [1] that,

$$f_{\text{av}}^{\text{BP}}=f_{\text{av}}^*+O(1/V),\quad \overline{q}^{\text{BP}}=O(V), \tag{12}$$

where $f_{\text{av}}^{\text{BP}}$ and $\overline{q}^{\text{BP}}$ are the expected average cost and the expected average network backlog size under Backpressure, respectively. Note that the performance results in (12) hold under Backpressure with *any* queueing discipline for choosing which packets to serve and for any V.

Though being a low-complexity technique that possesses wide applicability, the delay performance and the convergence speed of Backpressure are not satisfactory. Indeed, it is known that when Backpressure achieves a utility that is within $O(\epsilon)$ of the optimal, the average queueing delay is $\Theta(1/\epsilon)$. Although techniques proposed in [17] [18] are able to achieve an $O([\log(1/\epsilon)]^2)$ delay, it has been observed that the convergence time of these algorithms is $\Theta(1/\epsilon)$ (this will also be proven in Section 8).

Moreover, we make the following observation of Backpressure: it discards all past information of the system states, i.e., $\{S(0),...,S(t-1)\}$, and only reacts to the *instantaneous* state. While such an *incremental* manner is known to be able to accelerate the convergence of the algorithm compared to the ordinary subgradient methods [19], one interesting question to ask is whether such information can be utilized to construct better control techniques? Specifically, we are interested in understanding *how the information can be incorporated into algorithm design and whether this incorporation enables the development of algorithms that possess better delay and convergence performance.*

In our next section, we present two learning-aided control techniques that perform learning through the historic system state information. As we will see, this learning step allows us to achieve a near-optimal system performance and a significant improvement in convergence speed.

[2] A similar definition of Backpressure based on fluid model was also given in Section 4.8 of [20].

6. ONLINE LEARNING-AIDED CONTROL

In this section, we describe our online learning-aided system control idea and two novel control schemes, which we call Online Learning-Aided Control (OLAC) and OLAC2.

6.1 Intuition

Here we provide intuitions behind the two techniques. Both OLAC and OLAC2 are motivated by the fact that, under Backpressure algorithms, the queue vector plays the role of Lagrange multiplier [17]. However, Backpressure's incremental nature ignores the possibility of utilizing the information of the system dynamics for "accelerating" the convergence of the control algorithm, and only relies on taking subgradient-type updates. OLAC and OLAC2 are designed to simultaneously take both subgradient-type updates and statistical learning into consideration.

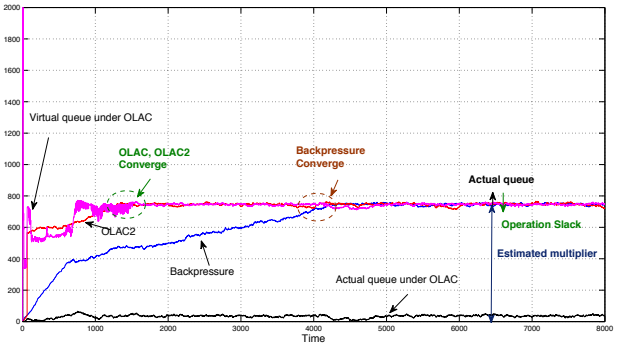

Figure 1: Convergence behavior of queue sizes under three algorithms, Backpressure, OLAC and OLAC2. It can be seen clearly that the virtual queue under OLAC and the actual queue under OLAC2 converge much faster to γ^* compared to the queue size under Backpressure. This clearly demonstrates the power of dual learning.

Fig. 1 best demonstrates our intuition and results. As we can see from the figure, the queue vector under Backpressure is attracted to the fixed point $\boldsymbol{\gamma}^*$ after some time [17]. However, Backpressure only uses the queue value to track the value $\boldsymbol{\gamma}^*$. This results in undesired delay and convergence performance. In OLAC and OLAC2, we instead introduce an auxiliary variable $\boldsymbol{\beta}(t)$ to approach the value of $\boldsymbol{\gamma}^*$ by solving an "empirical" version of (7) and finding the empirical Lagrange multiplier (called *dual learning*). The reason for performing this dual learning step is motivated by the fact that in many stochastic control problems, the optimal Lagrange multiplier $\boldsymbol{\gamma}^*$ is the unique key information for finding the optimal control policies. Then, we feed $\boldsymbol{\beta}(t)$ to the Backpressure controller for choosing the actions. Since $\boldsymbol{\beta}(t)$ eventually converges to $\boldsymbol{\gamma}^*$, we essentially do not require building up the queues for tracking $\boldsymbol{\gamma}^*$. Also, as we will see, at the beginning, dual learning provides a crude but fast learning of $\boldsymbol{\gamma}^*$. Hence, by doing so, we can significantly improve the convergence time of the system.

6.2 Dual Learning

The dual learning step is a novel and critical component in our online-learning-aided control mechanisms. Specifically, at every time t, the network operator maintains an empirical distribution of the network states, denoted by $\boldsymbol{\pi}(t) = (\pi_{s_1}(t), ..., \pi_{s_M}(t))$, where $\pi_{s_i}(t) = N_{s_i}(t)/t$ and $N_{s_i}(t)$ is the number of slots where $S(t) = s_i$ in $\{0, 1, ..., t-1\}$. Note that $\lim_{t\to\infty} \pi_{s_i}(t) = \pi_{s_i}$ $w.p.1$.

Although we adopt a simple learning mechanism for $\pi_{s_i}(t)$ here, other approaches for learning the distribution can also be used. In addition, if some prior knowledge of the system exists, it can be incorporated into $\pi_{s_i}(0)$, which further speeds up the learning. Other practical techniques in typical machine learning can also be included, such as using a uniform prior to avoid large fluctuation in the early stage of learning. Depending on the features of the approaches, similar performance and convergence results may also be obtained.

Using the empirical distribution, we then define the following empirical dual problem as follows:

$$\max: \ g(\boldsymbol{\beta}, t) \triangleq \sum_{s_i} \pi_{s_i}(t) g_{s_i}(\boldsymbol{\beta}), \quad \text{s.t.} \ \ \boldsymbol{\beta} \succeq \mathbf{0}. \tag{13}$$

We denote $\boldsymbol{\beta}(t) = (\beta_1^*(t), \beta_2^*(t), ..., \beta_r^*(t))^T$ an optimal solution vector of (13), and we call this step of obtaining $\boldsymbol{\beta}(t)$ via (13) *dual learning*. As we will see, $\boldsymbol{\beta}(t)$ plays a critical role in the proposed control schemes to significantly speed up the convergence to the optimal solution. On the other hand, by its definition, $\boldsymbol{\beta}(t)$ is time-varying and error-prone, which introduces significant challenges in the analysis of the proposed algorithms. Thus, novel techniques are developed in Sections 7 and 8 for performance analysis.

In the following, we present two control techniques using the empirical Lagrange multiplier $\boldsymbol{\beta}(t)$ obtained in dual learning for decision making.

6.3 OLAC

In the first technique, we use dual learning in each iteration of decision making. We also use a control parameter $\boldsymbol{\theta} > 0$ to manage the queue deviation (Its function and value will be specified later).

Online Learning-Aided Control (OLAC): At every time slot t, do:

- (Dual learning) Obtain $\boldsymbol{\beta}(t)$ by solving (13),
- (Action selection) Observe the system backlog $\boldsymbol{q}(t)$, and define the *effective* backlog $\boldsymbol{Q}(t)$ with:

$$Q_j(t) = q_j(t) + \beta_j(t) - \theta_j, \quad \forall\ j. \tag{14}$$

  Observe the current network state $S(t)$. If $S(t) = s_i$, choose $x^{(s_i)} \in \mathcal{X}^{(s_i)}$ that solves the following:

$$\max: \ -Vf(s_i, x) + \sum_{j=1}^{r} Q_j(t)\big[\mu_j(s_i, x) - A_j(s_i, x)\big] \tag{15}$$
$$\text{s.t.} \quad x \in \mathcal{X}^{(s_i)}.$$

- (Queueing update) Update all the queues with the arrival and service rates under the chosen actions according to (1). $\diamond$

We see that the OLAC algorithm has both similarities and differences compared to the Backpressure algorithm. On one hand, OLAC retains all the advantages of Backpressure, i.e., it does not require any statistical information of the system and it retains the low-complexity feature. On the other hand, one should note that OLAC has an important component that is not possessed by Backpressure, i.e., dual learning through empirical Lagrange multiplier. This step fundamentally changes the algorithm. In particular, $\boldsymbol{\beta}(t)$

takes advantage of the empirical information to speed up the convergence of $\boldsymbol{Q}(t)$ to the optimal Lagrange multiplier. In addition, it also serves the function of a "virtual" queue vector, and thus reduces the actual queue size of $\boldsymbol{q}(t)$ (See Fig. 1).

As we will see, OLAC achieves a near-optimal utility-delay tradeoff and has a significantly faster learning speed. The dual learning method also connects subgradient-type update analysis to the convergence properties of the system state distribution, allowing us to leverage useful results in the learning literature for performance analysis in stochastic network optimization.

A few comments on the control parameter $\boldsymbol{\theta} > 0$ are in order. As we will see later, the effective backlog $\boldsymbol{Q}(t)$ will eventually converge to $\boldsymbol{\gamma}^*$ (See the right side of Fig. 1). Since $\boldsymbol{\beta}(t)$ also converges to $\boldsymbol{\gamma}^*$, this implies that $\boldsymbol{\beta}(t)+\boldsymbol{q}(t)$ will eventually be larger than $\boldsymbol{\gamma}^*$. Thus, if we were to use only $\boldsymbol{\beta}(t)+\boldsymbol{q}(t)$ as the weight for the Backpressure controller, the system will always "think" that it is in a congested stage and operate at an "over-provisioned" mode, which will lead to utility loss. The introduction of $\boldsymbol{\theta}$ is trying to solve this problem. After subtracting $\boldsymbol{\theta}$, the effective backlog $\boldsymbol{Q}(t)$ can also go below $\boldsymbol{\gamma}^*$, which allows the Backpressure controller to choose "under-provisioned" actions and to achieve the desired performance through time-sharing over-provisioned and under-provisioned actions [21]. We will also show in Section 7 that it suffices to choose $\boldsymbol{\theta} = \Theta([\log(1/\epsilon)]^2)$ to guarantee an $O(\epsilon)$ close-to-optimal utility performance (Here $\epsilon = 1/V$).

6.4 OLAC2

We now present the second algorithm, which we call Online Learning-Aided Control 2 (OLAC2). OLAC2 combines the LIFO-Backpressure algorithm [18] with dual learning, and contains a backlog adjustment step.

Online Learning-Aided Control 2 (OLAC2): Choose a time $T_l = V^c$ for some $c \in [0, 1)$. At every time slot t, do:

- (Action selection) Observe the current network state $S(t)$ and queue backlog $\boldsymbol{q}(t)$. If $S(t) = s_i$, choose $x^{(s_i)} \in \mathcal{X}^{(s_i)}$ that solves the following:

$$\max: \ -Vf(s_i, x) + \sum_{j=1}^{r} q_j(t)\big[\mu_j(s_i, x) - A_j(s_i, x)\big] \quad (16)$$
$$\text{s.t.} \quad x \in \mathcal{X}^{(s_i)}.$$

- (Queueing update) Update all the queues with the arrival and service rates under the chosen actions according to (1) with the Last-In-First-Out (LIFO) discipline.

- (Dual learning and backlog adjustment) Only at $t = T_l$, solve (13) with $\boldsymbol{\pi}(T_l - 1)$ and obtain the empirical Lagrange multiplier $\tilde{\boldsymbol{\beta}}$. Then, make $\boldsymbol{q}(T_l) = \tilde{\boldsymbol{\beta}}$ by dropping packets if $q_j(T_l - 1) > \tilde{\beta}_j$ or adding null packets if $q_j(T_l - 1) < \tilde{\beta}_j$. $\diamond$

Note that although there exists a packet dropping step in dual learning and backlog adjustment, packet dropping rarely happens. The intuition is that at $t = T_l = V^c$, we have with high probability that $\tilde{\beta} = O(V)$. On the other hand, since the queue size increment is always $\Theta(1)$, we only have $\boldsymbol{q}(T_l - 1) = O(V^c)$. The key difference between OLAC2 and the LIFO-Backpressure algorithm developed in [18] is that OLAC2 contains a dual learning phase and adjusts its backlog condition at time $t = T_l$. The intuition here is that $\tilde{\boldsymbol{\beta}}$ computed at $t = T_l$ will give us good enough learning of the system, which provides better estimation of the true optimal Lagrangian multiplier than the queue value obtained via subgradient-type updates under LIFO-Backpressure. As we will see in Section 8 that, these two steps are critical for OLAC2, as they enable the achievement of a superior convergence performance.

Note that the LIFO discipline is important to OLAC2. This is so because if OLAC2 does not utilize any auxiliary process to "substitute" the true backlog. Thus, the queue will eventually build up, and its average value will still be $\boldsymbol{\gamma}^* = \Theta(V)$. Hence, packets can experience large delay. However, LIFO scheduling ensures that most of the packets are not affected by such network congestion and can go through the system with low delay.

6.5 Performance Guarantee

We summarize the proven properties of OLAC and OLAC2 here while leaving the technical details to next two sections.

OLAC: OLAC achieves the $[O(\epsilon), O([\log(1/\epsilon)]^2)]$ utility-delay tradeoff for general stochastic optimization problems. Compared to other Backpressure-based algorithms, OLAC is easier to implement and shows much better convergence performance (see Figure 1). The convergence analysis of OLAC is much more difficult because of the time-varying nature of $\boldsymbol{\beta}(t)$, and thus is left for future work.

OLAC2: OLAC2 achieves the $[O(\epsilon), O([\log(1/\epsilon)]^2)]$ utility-delay tradeoff for general stochastic optimization problems. In addition, it requires a convergence time of only $O(\epsilon^{-c} + \epsilon^{c/2-1}\log(1/\epsilon))$ with high probability. Therefore, by selecting $c = 2/3$ in OLAC2, we see that with very high probability, the system under OLAC2 will enter the near-optimal state in only $O(\epsilon^{-2/3}\log(1/\epsilon))$ time! This is in contrast to the Backpressure algorithm, whose convergence time is $\Theta(1/\epsilon)$. To the best of our knowledge, OLAC and OLAC2 are the first algorithms that simultaneously possess explicit near-optimal delay guarantee and sub-linear convergence time.

7. PERFORMANCE ANALYSIS

In this section, we first present some preliminaries needed for our analysis. Then, we present the detailed performance results for OLAC and OLAC2.

7.1 The augmented problem and preliminaries

Due to the introduction of $\boldsymbol{\beta}(t)$ in decision making, the performance of OLAC and OLAC2 cannot be obtained by directly applying the typical Lyapunov analysis. To overcome this obstacle, we introduce the following *augmented problem* and carry out our analysis based on it. Specifically, we define:

$$\min: F(\boldsymbol{x}) + \sum_j (\beta_j(t) - \theta_j) H_j(\boldsymbol{x}) \quad (17)$$
$$\text{s.t.} \ H_j(\boldsymbol{x}) \le 0, \ \forall\, j,$$
$$x^{(s_i)} \in \mathcal{X}^{(s_i)} \quad \forall\, i = 1, 2, ..., M.$$

Let $\tilde{g}(\boldsymbol{\gamma})$ be the dual function of (17), i.e.,

$$\tilde{g}(\boldsymbol{\gamma}) = \inf_{x^{(s_i)} \in \mathcal{X}^{(s_i)}} \sum_{s_i} \pi_{s_i} \Big\{ Vf(s_i, x^{(s_i)}) \quad (18)$$

$$+\sum_j[\gamma_j+\beta_j(t)-\theta_j]\big[A_j(s_i,x^{(s_i)})-\mu_j(s_i,x^{(s_i)})\big]\Big\}.$$

It is interesting to notice the similarity between (18) and (15) (note that $Q_j(t)=q_j(t)+\beta_j(t)-\theta_j$). Using the definition of $g(\boldsymbol{\gamma})$, we have:

$$\tilde{g}(\boldsymbol{\gamma})=g(\boldsymbol{\gamma}+\boldsymbol{\beta}(t)-\boldsymbol{\theta}). \quad (19)$$

In the following, we state the assumptions we make throughout the paper. These assumptions are mild and can typically be satisfied in network optimization problems.

Assumption 1. *There exists a constant $\epsilon_s=\Theta(1)>0$ such that for any valid state distribution $\boldsymbol{\pi}'=(\pi'_{s_1},...,\pi'_{s_M})$ with $\|\boldsymbol{\pi}'-\boldsymbol{\pi}\|\leq\epsilon_s$, there exists a set of actions $\{x_k^{(s_i)}\}_{i=1,...,M}^{k=1,2,...,\infty}$ with $x_k^{(s_i)}\in\mathcal{X}^{(s_i)}$ and some variables $\vartheta_k^{(s_i)}\geq 0$ for all s_i and k with $\sum_k\vartheta_k^{(s_i)}=1$ for all s_i (possibly depending on $\boldsymbol{\pi}'$), such that:*

$$\sum_{s_i}\pi_{s_i}\Big\{\sum_k\vartheta_k^{(s_i)}[A_j(s_i,x_k^{(s_i)})-\mu_j(s_i,x_k^{(s_i)})]\Big\}\leq-\eta_0, \quad (20)$$

where $\eta_0=\Theta(1)>0$ is independent of $\boldsymbol{\pi}'$. $\diamond$

Assumption 2. *There exists a set of actions $\{x_k^{(s_i)*}\}_{i=1,...,M}^{k=1,...,\infty}$ with $x_k^{(s_i)*}\in\mathcal{X}^{(s_i)}$ and some variables $\vartheta_k^{(s_i)*}\geq 0$ for all s_i and k with $\sum_k\vartheta_k^{(s_i)*}=1$ for all s_i, such that:*

$$\sum_{s_i}\pi_{s_i}\sum_k\vartheta_k^{(s_i)*}f(s_i,x_k^{(s_i)*})=f^*_{av}, \quad (21)$$

$$\sum_{s_i}\pi_{s_i}\Big\{\sum_k\vartheta_k^{(s_i)*}[A_j(s_i,x_k^{(s_i)*})-\mu_j(s_i,x_k^{(s_i)*})]\Big\}=0.$$ $\diamond$

Assumption 3. *$\boldsymbol{\gamma}_0^*$ is the unique optimal solution of $g_0(\boldsymbol{\gamma})$ in $\mathbb{R}^r$.* $\diamond$

Some remarks are in order. Notice that by having $\eta_0>0$ in Assumption 1, we assume that there exists at least one control policy under which the resulting system arrival rate vector is strictly smaller than the resulting service rate vector (or the arrival rate vector is strictly inside the capacity region if it is exogenous). This is known as the "slack" condition, and is commonly made in the literature with $\epsilon_s=0$, e.g., [22], [23] (it is always necessary to have $\eta_0\geq 0$ for system stability [1]). Here with $\epsilon_s>0$, we assume in addition that when two systems are relatively "similar" to each other, they can both be stabilized by some randomized control policy (may be different) that results in the same slack. Assumption 2 is also commonly satisfied by most network optimization problems, especially when the cost $f(s_i,x^{(s_i)})$ increases with the increment of the services rate $\mu_j(s_i,x^{(s_i)})$. Finally, Assumption 3 holds for many network utility optimization problems, especially when the corresponding cost functions are strictly convex, e.g., [2] and [17].

Under these assumptions, our first lemma shows that the magnitude of $\boldsymbol{\beta}(t)$ quickly becomes bounded as time goes on.

Lemma 1. *There exists an $O(1)$ time $T_{\epsilon_s}<\infty$, such that with probability 1, for all $t\geq T_{\epsilon_s}$,*

$$\sum_j\beta_j(t)\leq\xi\triangleq\frac{Vf_{\max}}{\eta_0}.$$ $\diamond$

Proof. See Appendix A. $\square$

With the above bound, we have the following corollary, which shows the convergence of $g(\boldsymbol{\beta},t)$ to $g(\boldsymbol{\beta})$.

Corollary 1. *With probability 1, for all $t\geq T_{\epsilon_s}$ (here T_{ϵ_s} is defined in Lemma 1), the function $g(\boldsymbol{\beta},t)$ satisfies:*

$$|g(\boldsymbol{\beta}(t),t)-g(\boldsymbol{\beta}(t))|\leq\max_{s_i}|\delta_{s_i}(t)|M(Vf_{\max}+r\xi B).$$

Here $\xi=\frac{Vf_{\max}}{\eta_0}$ is defined in Lemma 1, $\delta_{s_i}(t)\triangleq\pi_{s_i}-\pi_{s_i}(t)$ is the estimation error of the empirical distribution for state s_i, and M is the number of system states. $\diamond$

Proof. See Appendix B. $\square$

With Lemma 1 and Corollary 1, we next show that $\boldsymbol{\beta}(t)$ converges to $\boldsymbol{\gamma}^*$ with probability 1.

Lemma 2. $\lim_{t\to\infty}\boldsymbol{\beta}(t)=\boldsymbol{\gamma}^*$ *w.p.*1. $\diamond$

Proof. Omitted due to space limitation. See [24] for details. $\square$

Notice that while Assumption 3 assumes that there is a unique optimal for (4), it does not guarantee that $g(\boldsymbol{\beta},t)$ will also only have a unique solution. Lemma 2 thus shows that although such a condition can appear, it is simply a transient phenomenon and $\boldsymbol{\beta}(t)$ will eventually converge to $\boldsymbol{\gamma}^*$. According to Assumption 3, (19) implies that the unique maximizer of $\tilde{g}(\boldsymbol{\gamma})$, denoted by $\tilde{\boldsymbol{\gamma}}^*(t)$, is given by $\tilde{\boldsymbol{\gamma}}^*(t)=\boldsymbol{\gamma}^*-\boldsymbol{\beta}(t)+\boldsymbol{\theta}$. Using Lemma 2, we also see that:

$$\lim_{t\to\infty}\tilde{\boldsymbol{\gamma}}^*(t)\to\boldsymbol{\theta}, \quad w.p.1. \quad (22)$$

In the following, we will carry out our analysis about the performance of `OLAC` and `OLAC2` under a general system structure that commonly appears in practice. For our analysis, it is also useful to define $B\triangleq\frac{r}{2}\delta_{\max}^2$. It can be seen that $\|\boldsymbol{\mu}(t)-\boldsymbol{A}(t)\|\leq B$ at all time.

7.2 Performance of OLAC and OLAC2

We now carry out our analysis for `OLAC` and `OLAC2` under a *polyhedral* system structure. Intuitively speaking, this structure appears in systems where the feasible control action sets are finite, in which case once an action becomes the minimizer of the dual function at $\boldsymbol{\gamma}$, it remains so at all $\boldsymbol{\gamma}'$ in some neighborhood of $\boldsymbol{\gamma}$, resulting in a constant service and arrival rate difference (which determines the slope of $g(\boldsymbol{\gamma})$). The formal definition of the polyhedral structure is as follows [17].

Definition 1. *A system is polyhedral with parameter $\rho>0$ if the dual function $g_0(\boldsymbol{\gamma})$ satisfies:*

$$g_0(\boldsymbol{\gamma}_0^*)\geq g_0(\boldsymbol{\gamma})+\rho\|\boldsymbol{\gamma}_0^*-\boldsymbol{\gamma}\|.$$ $\diamond$ $\quad (23)$

Using (23), we have:

$$Vg_0(\boldsymbol{\gamma}_0^*)\geq Vg_0(\boldsymbol{\gamma}/V)+V\rho\|\boldsymbol{\gamma}_0^*-\boldsymbol{\gamma}/V\|.$$

Together with (8) and the fact that $\boldsymbol{\gamma}^*=V\boldsymbol{\gamma}_0^*$, we see that the above implies:

$$g(\boldsymbol{\gamma}^*)\geq g(\boldsymbol{\gamma})+\rho\|\boldsymbol{\gamma}^*-\boldsymbol{\gamma}\|, \quad (24)$$

i.e., the function $g(\boldsymbol{\gamma})$ also satisfies the polyhedral condition (23) with the same parameter ρ. Moreover, we have:

$$\begin{aligned}\tilde{g}(\tilde{\boldsymbol{\gamma}}^*(t)) &= g(\tilde{\boldsymbol{\gamma}}^*(t)+\boldsymbol{\beta}(t)-\boldsymbol{\theta})\\ &\overset{(a)}{\geq} g(\boldsymbol{\gamma}+\boldsymbol{\beta}(t)-\boldsymbol{\theta})+\rho\|\tilde{\boldsymbol{\gamma}}^*(t)-\boldsymbol{\gamma}\|\end{aligned}$$

$$= \tilde{g}(\boldsymbol{\gamma}) + \rho\|\tilde{\boldsymbol{\gamma}}^*(t) - \boldsymbol{\gamma}\|. \quad (25)$$

Here (a) is because $\boldsymbol{\gamma}^* = \tilde{\boldsymbol{\gamma}}^*(t) + \boldsymbol{\beta} - \boldsymbol{\theta}$. This shows that the function $\tilde{g}(\boldsymbol{\gamma})$ is also polyhedral. It turns out that this polyhedral condition has a special attraction property, under which the queue vector under OLAC and OLAC2 will be exponentially attracted towards $\tilde{\boldsymbol{\gamma}}^*(t)$. This is shown in the following important lemma:

Lemma 3. *Suppose (23) holds. Then, there exist a constant* $D_p \triangleq \frac{B-\eta^2}{2(\rho-\eta)} = \Theta(1)$ *with* $\eta < \rho$, *and a finite time* $T_0 < \infty$, *such that, with probability* 1, *for all* $t \geq T_0$, *if* $\|\boldsymbol{q}(t) - \tilde{\boldsymbol{\gamma}}^*(t)\| \geq D$,

$$\mathbb{E}\big\{\|\boldsymbol{q}(t+1) - \tilde{\boldsymbol{\gamma}}^*(t)\| \mid \boldsymbol{q}(t)\big\} \leq \|\boldsymbol{q}(t) - \tilde{\boldsymbol{\gamma}}^*(t)\| - \eta. \ \diamond \quad (26)$$

PROOF. Omitted due to space limitation. See [24] for details. □

Lemma 3 states that, after some finite time T_0, if the queue vector $\boldsymbol{q}(t)$ is of distance $D = \Theta(1)$ away from $\tilde{\boldsymbol{\gamma}}^*(t)$, there will be an $\Theta(1)$ drift "pushing" the queue size towards $\tilde{\boldsymbol{\gamma}}^*(t)$. This intuitively explains why $\boldsymbol{q}(t)$ mostly stays close to a fixed point. Note that proof of Lemma 3 is based on the augmented problem (17). Below, we present the detailed performance results. Recall that all the results are obtained under Assumptions 1, 2, and 3.

7.2.1 Performance of OLAC

We now analyze the performance of the OLAC algorithm. It is important to note that $\tilde{\boldsymbol{\gamma}}^*(t)$ is a function of $\boldsymbol{\beta}(t)$. Under OLAC, however, the value of $\boldsymbol{\beta}(t)$ changes every time slot. Hence, analyzing the performance of OLAC is non-trivial. Fortunately, using (22), we know that $\tilde{\boldsymbol{\gamma}}^*(t)$ eventually converges to $\boldsymbol{\theta}$ with probability 1. This allows us to prove the following theorem, which states that the average queue size under OLAC is mainly determined by the vector $\boldsymbol{\theta}$.

Theorem 1. *Suppose (i)* $g_0(\boldsymbol{\gamma})$ *is polyhedral with* $\rho = \Theta(1) > 0$, *and (ii)* $\boldsymbol{Q}(t)$ *has a countable state space under* OLAC. *Then, under* OLAC, *we have w.p.*1 *that:*

$$\overline{q}_{av} = \sum_j \theta_j + O(1). \quad (27)$$

PROOF. Omitted due to space limitation. See [24] for details. □

Theorem 1 shows that by using the decision making rule (15) with the effective backlog $\boldsymbol{Q}(t)$, one can guarantee that the average queue size is roughly $\sum_j \theta_j$. Hence, by choosing $\theta_j = \Theta([\log(V)]^2)$, we recover the delay performance achieved in [17] and [18].

It is tempting to choose $\theta_j = 0$, in which case one can indeed make the average queue size $\Theta(1)$. However, as discussed before, the choice of $\boldsymbol{\theta}$ also affects the utility performance of OLAC. Indeed, choosing a small $\boldsymbol{\theta}$ forces the system to run in an *over-provision* mode by always having an effectively backlog very close (or larger) to $\boldsymbol{\gamma}^*$. However, in order to achieve a near-optimal utility performance, the control algorithm must be able to timeshare the over-provision mode and under-provision actions [21]. Therefore, it is necessary to use a larger $\boldsymbol{\theta}$ value.

In the following theorem, we show that an $O(1/V)$ close-to-optimal utility can be achieved as long as we choose $\theta_j = \Theta([\log(V)]^2)$ for all $j = 1, ..., r$. Our analysis is different from the previous Lyapunov analysis and will be useful for analyzing similar problems (Recall that $\epsilon = 1/V$).

Theorem 2. *Suppose the conditions in Theorem 1 hold. Then, with a sufficiently large* V *and* $\theta_j = [\log(V)]^2$ *for all* j, *we have w.p.*1 *that:*

$$f_{av}^{\texttt{OLAC}} = f_{av}^* + O(\frac{1}{V}). \quad (28)$$

PROOF. See Appendix C. □

Theorems 1 and 2 together show that OLAC achieves the $[O(1/V), O([\log(V)]^2)]$ utility-delay tradeoff for general stochastic optimization problems. Compared to the algorithms developed in [17] [18], which also achieve the same tradeoff, OLAC preserves using the First-In-First-Out (FIFO) queueing discipline and does not require any pre-learning phase. Thus, it is more suitable for practical implementations.

7.2.2 Performance of OLAC2

We now present the performance results of OLAC2. Since OLAC2 is equivalent to LIFO-backpressure [18] except for the learning step and a finite backlog adjustment at time $t = T_l$, its performance is the same as LIFO-Backpressure. [3]

The following theorem from [18] summarizes the results.

Theorem 3. *Suppose (i)* $g_0(\boldsymbol{\gamma})$ *is polyhedral with* $\rho = \Theta(1) > 0$, *and (ii)* $\boldsymbol{q}(t)$ *has a countable state space under* OLAC2. *Then, with a sufficiently large* V,

1) *The utility under* OLAC2 *satisfies* $f_{av}^{\texttt{OLAC2}} = f_{av}^* + O(\frac{1}{V})$.

2) *For any queue* j *with a time average input rate* $\lambda_j > 0$, *there are a subset of packets from the arrivals that eventually depart the queue and have an average rate* $\tilde{\lambda}_j$ *that satisfies:*

$$\lambda_j \geq \tilde{\lambda}_j \geq \left[\lambda_j - O(\frac{1}{V^{\log(V)}})\right]^+. \quad (29)$$

Moreover, the average delay of these packets is $O(\frac{[\log(V)]^2}{\tilde{\lambda}_j})$.

PROOF. See [18]. □

Theorem 3 shows that if a queue $q_j(t)$ has an input rate $\lambda_j = \Theta(1)$, then, under OLAC2, almost all the packets going through $q_j(t)$ will experience only $O([\log(V)]^2)$ delay. Applying this argument to every queue in the network, we see that the average network delay is roughly $O(r[\log(V)]^2)$.

8. CONVERGENCE TIME ANALYSIS

In this section, we study another important performance metric of our techniques, the convergence time. Convergence time measures how fast an algorithm reaches its steady-state. Hence, it is an important indicator of the robustness of the technique. However, it turns out that due to the complex nature of the $\boldsymbol{\beta}(t)$ process and the structure of the systems, the convergence properties of the algorithms are hard to analyze. Therefore, we mainly focus on analyzing the convergence time of OLAC2. We now give the formal definition of the convergence time of an algorithm. In the definition, we use $\boldsymbol{\gamma}(t)$ to denote the estimated Lagrange multiplier under the control schemes, e.g., $\boldsymbol{q}(t)$ under Backpressure and OLAC2 and $\boldsymbol{\beta}(t) - \boldsymbol{\theta} + \boldsymbol{q}(t)$ under OLAC.

[3]Note that OLAC2 may need to discard packets in the backlog dropping step. However, such an event happens with very small probability, as can be seen in the proof of Theorem 5.

Definition 2. *Let $\zeta > 0$ be a given constant. The ζ-convergence time of the control algorithm, denoted by T_ζ, is the time it takes for the estimated Lagrange multiplier $\boldsymbol{\gamma}(t)$ to get to within ζ distance of $\boldsymbol{\gamma}^*$, i.e.,*

$$T_\zeta \triangleq \inf\{t \,|\, ||\boldsymbol{\gamma}(t) - \boldsymbol{\gamma}^*|| \leq \zeta\}. \ \diamond \tag{30}$$

Note that our definition is different from the definition in [25], where convergence time relates to how fast the time-average rates converge to the optimal values. In our case, the convergence time definition (30) is motivated by the fact that Backpressure, `OLAC`, and `OLAC2` all use functions of the queue vector to estimate $\boldsymbol{\gamma}^*$, which is the key for determining the optimal control actions. Hence, the faster the algorithm learns $\boldsymbol{\gamma}^*$, the faster the system enters the optimal operating zone.

In the following, we present the convergence results of `OLAC2`. As a benchmark comparison, we first study the convergence time of the Backpressure algorithm. Since both Backpressure and `OLAC2` use the actual queue size as the estimate of $\boldsymbol{\gamma}^*$, we will analyze the convergence time with $\boldsymbol{\gamma}(t) = \boldsymbol{q}(t)$.

8.1 Convergence time of Backpressure

We start by analyzing the convergence time of Backpressure. Our result is summarized in the following theorem.

Theorem 4. *Suppose that $g_0(\boldsymbol{\gamma})$ is polyhedral with parameter $\rho = \Theta(1) > 0$. Then,*

$$\big(\|\boldsymbol{q}(0) - \boldsymbol{\gamma}^*\| - D_p\big)^+/B \leq \mathbb{E}\big\{T_{D_p}\big\} \leq \|\boldsymbol{q}(0) - \boldsymbol{\gamma}^*\|/\eta. \tag{31}$$

Here $\eta \in [0, \rho) = \Theta(1)$ and $D_p = \frac{B-\eta^2}{2(\rho-\eta)} = \Theta(1)$. $\diamond$

Proof. See Appendix D. $\square$

Since $\boldsymbol{\gamma}^* = \Theta(V)$, if $\boldsymbol{q}(0) = \Theta(1)$, e.g., $\boldsymbol{q}(0) = \mathbf{0}$, Theorem 4 states that the expected convergence time of Backpressure is $\Theta(V)$. This is consistent with the results that have been observed in previous works, e.g., [1] and [17].

8.2 Convergence time of OLAC2

We now study the convergence time of the `OLAC2` scheme. Notice that under `OLAC2` the convergence of $\boldsymbol{q}(t)$ to $\boldsymbol{\gamma}^*$ depends on both $\tilde{\boldsymbol{\beta}}$ and the dynamics of the queue vector $\boldsymbol{q}(t)$. The following theorem characterizes the convergence time of `OLAC2`.

Theorem 5. *Suppose (i) $g_0(\boldsymbol{\gamma})$ is polyhedral with $\rho = \Theta(1) > 0$, and (ii) $\boldsymbol{q}(t)$ has a countable state space under `OLAC2`. Then, with a sufficiently large V, we have with probability of at least $1 - \frac{M}{V^{4\log(V)}}$ that, under `OLAC2`,*

$$\mathbb{E}\big\{T_{D_p}\big\} = O(V^{1-c/2}\log(V) + V^c). \quad \diamond \tag{32}$$

Proof. See Appendix E. $\square$

Choosing $c = \frac{2}{3}$, we notice that with very high probability, the system under `OLAC2` will enter the near-optimal state in only $O(V^{2/3}\log(V))$ time! This is in contrast to the Backpressure algorithm, whose convergence time is $\Theta(V)$ as shown in Theorem 4.

9. SIMULATION

In this section, we provide simulation results for `OLAC` and `OLAC2` to demonstrate both the near-optimal performance and the the superior convergence rate. We consider a 2-queue system depicted in Fig. 2.

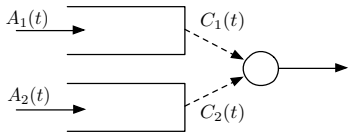

Figure 2: A 2-queue system. In this system, each queue receives random arrivals. The server can only serve one queue at a time.

Here $A_j(t)$ denotes the number of arriving packets to queue j at time t. We assume that $A_j(t)$ is i.i.d. with either 2 or 0 with probabilities p_j and $1-p_j$, where $p_1 = 0.3$ and $p_2 = 0.4$. In this case, we see that $\lambda_1 = 0.6$ and $\lambda_2 = 0.8$. We assume that each queue has a time-varying channel and denote its state by $C_j(t)$. $C_j(t)$ takes value in the set $\mathcal{C} = \{0, 2, 4, 6\}$. At each time, the server decides how much power to allocate to each queue. We denote $P_j(t)$ the power allocated to queue j at time t. Then, the instantaneous service rate a queue obtains is given by:

$$\mu_j(t) = \log(1 + C_j(t)P_j(t)). \tag{33}$$

The feasible power allocation set is given by $\mathcal{P} = \{0, 0.75, 1.5, 2.25, 3\}$. The objective is to stabilize the system with minimum average power. It is important to note that, even though this is a simple setting, it is actually quite representative and models many problems in different contexts, e.g., a downlink system in wireless network, workload scheduling in a power system, inventory control system, and traffic light control. Moreover, it can be verified that Assumptions 1, 2 and 3 all hold in this example.

We compare three algorithms, Backpressure, `OLAC` and `OLAC2`. We also consider two different channel distributions. In the first distribution, $C_j(t)$ takes each value in $\mathcal{C}$ with probability 0.25, whereas in the second distribution, $C_j(t)$ takes values 0 and 6 with probability 0.1 and takes values 2 and 4 with probability 0.4. This is to mimic the Gaussian distribution. Note that in each case, we have a total of 16 different channel state combinations.

Fig. 3 and 4 show that `OLAC` and `OLAC2` significantly outperform the Backpressure in terms of delay. For example, in the uniform channel case, we can see from Fig. 3 that when $V = 100$, the average power performance is indistinguishable under the three algorithms. Backpressure results in an average delay of 210 slots while `OLAC` and `OLAC2` only incur an average delay about 20 slots, which is 10 *times* smaller! Fig. 4 shows similar behavior of the algorithms under the unbalanced channel distribution.

Having seen the utility-delay tradeoff of the algorithms, we now look at the convergence rate of the algorithms. Fig. 1 shows the convergence behavior of the first queue under the three different algorithms under the uniform channel distribution with $V = 500$. As we can see, the virtual queue size under `OLAC` and the actual queue size under `OLAC2` converge very quickly. Compared to Backpressure, the convergence time is reduced by about 2500 timeslots. The behavior of the other queue is similar and hence we do not present the results. It is important to notice here that since `OLAC` and `OLAC2` converge very quickly, the early arrivals into the system will depart from the queue without much waiting time (see the queue process under `OLAC`), whereas in Backpres-

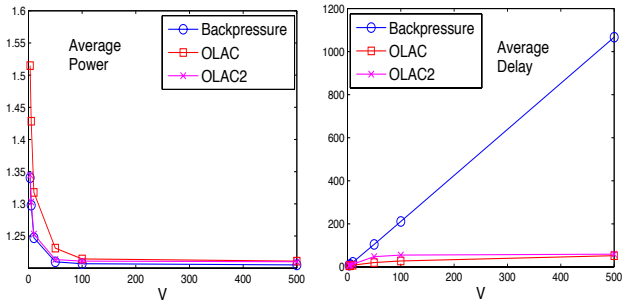

Figure 3: The power-delay performance of the algorithms under uniform channel distribution $[0.25, 0.25, 0.25, 0.25]$. We see that although all algorithms achieve similar near-minimum average power, OLAC and OLAC2 achieve this performance with much smaller delay.

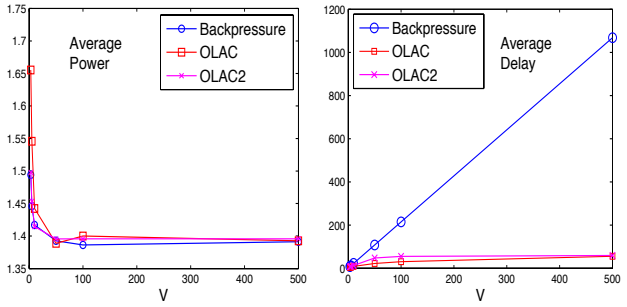

Figure 4: The power-delay performance of the algorithms under the unbalanced channel distribution $[0.1, 0.4, 0.4, 0.1]$. We see again that OLAC and OLAC2 significantly outperform Backpressure in delay.

sure, early arrivals must wait until the algorithm converge to get serve and suffer from a long delay. Fig. 5 also shows the "jump" behavior of the sizes of the queues under OLAC2 under the uniform channel distribution and $V = 500$. It can be seen that the convergence time is significantly reduced via such dual learning (by about 2500 timeslots).

10. CONCLUSION

In this paper, we take a first step to investigate the power of online learning in stochastic network optimization with unknown system statistics *a priori*. We propose two learning-aided control techniques OLAC and OLAC2 for controlling stochastic queueing systems. The design of OLAC and OLAC2 incorporates statistical learning into system control, and provides new ways for designing algorithms for achieving near-optimal system performance for general system utility maximization problems. Moreover, incorporating the learning step significantly improves the convergence speed of the control algorithms. Such insights are not only proven via novel analytical techniques, but also validated through numerical simulations. Our study demonstrates the promising power of online learning in stochastic network optimization. Much further investigation is desired. In particular, we would like to provide theoretical guarantees on the convergence speed on OLAC, and more importantly, understand the fundamental limit of convergence in the online-learning-aided stochastic network optimization. We would like to further investigate practical application scenarios of the proposed schemes and address the corresponding challenges in real world applications.

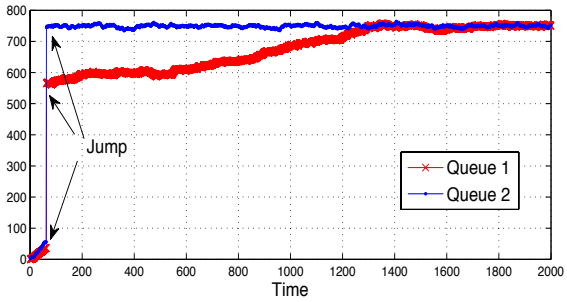

Figure 5: Convergence of the queues to the optimal Lagrange multiplier under OLAC2 with uniform channel distribution and $V = 500$. It can be seen that after only around 80 slots, dual learning already generates a very good estimation of γ^*. This significantly reduces the time for convergence.

11. ACKNOWLEDGEMENT

This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300,2011CBA00301, the National Natural Science Foundation of China Grant 61033001, 61361136003, 61303195, the China youth 1000-talent grant, and the MSRA-Tsinghua Joint lab grant.

12. REFERENCES

[1] L. Georgiadis, M. J. Neely, and L. Tassiulas. *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking Vol. 1, no. 1, pp. 1-144, 2006.

[2] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. *IEEE/ACM Trans. Netw.*, 15(6):1333–1344, 2007.

[3] R. Urgaonkar and M. J. Neely. Opportunistic scheduling with reliability guarantees in cognitive radio networks. *IEEE Transactions on Mobile Computing*, 8(6):766–777, June 2009.

[4] L. Huang, J. Walrand, and K. Ramchandran. Optimal smart grid tariff. *Information Theory and Applications Workshop (ITA) (Invited), San Diego*, Feb 2012.

[5] M. J. Neely, A. S. Tehrani, and A. G. Dimakis. Effficient algorithms for renewable energy allocation to delay tolerant consumers. *Proceedings of IEEE SmartGridComm*, Oct 2010.

[6] M. J. Neely R. Urgaonkar, B. Urgaonkar and A. Sivasubramaniam. Optimal power cost management using stored energy in data centers. *Proceedings of ACM Sigmetrics*, June 2011.

[7] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vols. I and II*. Boston: Athena Scientific, 2005 and 2007.

[8] M. J. Neely and L. Huang. Dynamic product assembly and inventory control for maximum profit. *IEEE Conference on Decision and Control (CDC), Atlanta, Georgia*, Dec. 2010.

[9] J. G. Dai and W. Lin. Maximum pressure policies in stochastic processing networks. *Operations Research*, 53(2):197–218, March-April 2005.

[10] L. Jiang and J. Walrand. Stable and utility-maximizing scheduling for stochastic processing networks. *Allerton Conference on Communication, Control, and Computing*, 2009.

[11] P. Varaiya. *The Max-Pressure Controller for Arbitrary Networks of Signalized Intersections*. Advances in Dynamic Network Modeling in Complex 27 Transportation Systems, Complex Networks and Dynamic Systems 2, Springer Science+Business Media New York, 2013.

[12] T. Le, P. Kovacs, N. Walton, H. L. Vu, L. Andrew, and S. Hoogendoorn. Decentralized signal control for urban road networks. *ArXiv Technical Teport, arXiv:1310.0491v1*, Oct 2013.

[13] S. H. Low and D. E. Lapsley. Optimization flow control-I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 7(6):861–875, Dec. 1999.

[14] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, January 2007.

[15] L. Huang and M. J. Neely. The optimality of two prices: Maximizing revenue in a stochastic network. *IEEE/ACM Transactions on Networking*, 18(2):406–419, April 2010.

[16] M. J. Neely. Super-fast delay tradeoffs for utility optimal fair scheduling in wireless networks. *IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Nonlinear Optimization of Communication Systems*, 24(8):1489–1501, Aug. 2006.

[17] L. Huang and M. J. Neely. Delay reduction via Lagrange multipliers in stochastic network optimization. *IEEE Trans. on Automatic Control*, 56(4):842–857, April 2011.

[18] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari. LIFO-backpressure achieves near optimal utility-delay tradeoff. *IEEE/ACM Transactions on Networking*, 21(3):831–844, June 2013.

[19] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Boston: Athena Scientific, 2003.

[20] Sean Meyn. *Control Techniques for Complex Networks*. Cambridge University Press, 2007.

[21] M. J. Neely. Optimal energy and delay tradeoffs for multi-user wireless downlinks. *IEEE Transactions on Information Theory*, 53(9):3095–3113, Sept. 2007.

[22] L. Ying, S. Shakkottai, and A. Reddy. On combining shortest-path and back-pressure routing over multihop wireless networks. *Proceedings of IEEE INFOCOM*, April 2009.

[23] L. Bui, R. Srikant, and A. Stolyar. Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing. *Proceedings of IEEE INFOCOM Mini-Conference*, April 2009.

[24] L. Huang, X. Liu, and X. Hao. The power of online learning in stochastic network optimization. *arXiv Technical Report, arXiv:1404.1592*, 2014.

[25] B. Li, A. Eryilmaz, and R. Li. Wireless scheduling for utility maximization with optimal convergence speed. *Proceedings of IEEE INFOCOM, Turin, Italy*, April 2013.

[26] R. Durrett. *Probability: Theory and Examples*. Duxbury Press, 3rd edition, 2004.

[27] L. Huang and M. J. Neely. Max-weight achieves the exact $[O(1/V), O(V)]$ utility-delay tradeoff under Markov dynamics. *arXiv:1008.0200v1*, 2010.

[28] F. Chung and L. Lu. Concentration inequalities and martingale inequalities - a survey. *Internet Math.*, 3 (2006-2007), 79–127.

Appendix A – Proof of Lemma 1

We prove Lemma 1 here.

PROOF. (Lemma 1) First we see that $\pi_{s_i}(t)$ converges to π_{s_i} as t goes to infinity with probability 1 [26]. Thus, there exists a time $T_{\epsilon_s} < \infty$, such that $\|\boldsymbol{\pi}(t) - \boldsymbol{\pi}\| \leq \epsilon_s$ for all $t \geq T_{\epsilon_s}$ with probability 1. This implies that $\boldsymbol{\beta}(t) < \infty$ for all $t \geq T_{\epsilon_s}$ [19]. Indeed, under Assumption 1, we see that there exists an $\eta_0 > 0$ such that (20) holds.

We now construct a *fictitious* system, which is exactly the same as our system, except that we replace the network state distribution $\boldsymbol{\pi}$ by $\boldsymbol{\pi}(t)$. From Assumption 1, we know that for any $t \geq T_{\epsilon_s}$, $w.p.1$, this system admits an optimal control policy that achieves the optimal cost (with the state distribution being $\boldsymbol{\pi}(t)$) and ensures network stability [1]. We denote $f^*_{\text{av}}(\boldsymbol{\pi}(t))$ the optimal average cost of the fictitious system subject to stability. We see then $f^*_{\text{av}}(\boldsymbol{\pi}(t)) \geq 0$.

Now we apply Theorem 1 in [27] to get that $g(\boldsymbol{\beta}(t), t) = f^*_{\text{av}}(\boldsymbol{\pi}(t))$. Therefore, for any $t \geq T_{\epsilon_s}$, one can plug the variables that result in the slack in (20) into $g(\boldsymbol{\beta}, t)$ and obtain:

$$\begin{aligned} 0 &\leq f^*_{\text{av}}(\boldsymbol{\pi}(t)) \\ &\overset{(a)}{=} g(\boldsymbol{\beta}(t), t) \\ &\overset{(b)}{\leq} \sum_{s_i} \pi_{s_i}(t)[V f_{\max} - \eta_0 \sum_j \beta_j(t)]. \end{aligned}$$

Here (a) follows from Theorem 1 in [27], and (b) follows from the definition of $g(\boldsymbol{\beta}, t)$. This shows that $w.p.1$,

$$\sum_j \beta_j(t) \leq \xi \triangleq \frac{V f_{\max}}{\eta_0}, \ \forall\, t \geq T_{\epsilon_s}. \tag{34}$$

This proves Lemma 1. □

Appendix B – Proof of Corollary 1

We carry out our proof for Corollary 1 here.

PROOF. (Corollary 1) From lemma 1, we see that $w.p.1$ after some T_{ϵ_s} time, $\|\boldsymbol{\beta}(t)\| \leq \xi$. This implies that, for all $t \geq T_{\epsilon_s}$ and all s_i, we have $w.p.1$ that:

$$g_{s_i}(\boldsymbol{\beta}(t)) \leq V f_{\max} + r\xi B, \quad \forall\, s_i. \tag{35}$$

Now note that $g(\boldsymbol{\beta}(t), t)$ can be expressed as:

$$g(\boldsymbol{\beta}(t), t) = \sum_{s_i} [\pi_{s_i} + \delta_{s_i}(t)] g_{s_i}(\boldsymbol{\beta}(t)), \tag{36}$$

where $\delta_{s_i}(t) = \pi_{s_i}(t) - \pi_{s_i}$ denotes the error of the empirical distribution. Therefore,

$$|g(\boldsymbol{\beta}(t), t) - g(\boldsymbol{\beta}(t))| \leq \max_{s_i} |\delta_{s_i}(t)| \sum_{s_i} |g_{s_i}(\boldsymbol{\beta}(t))| \tag{37}$$

$$\overset{(a)}{\leq} \max_{s_i} |\delta_{s_i}(t)| M(V f_{\max} + r\xi B).$$

The inequality (a) uses the fact that $g_{s_i}(\boldsymbol{\beta}(t)) \leq V f_{\max} + r\xi B$ for all $t \geq T_{\epsilon_s}$. This implies that with probability 1, after some finite time T_{ϵ_s}, (37) holds for all points $\boldsymbol{\beta}(t)$ generated by solving (13). $\square$

Appendix C – Proof of Theorem 2

To prove Theorem 2, we first have the following simple lemma, which will be used in the proof.

Lemma 4. *Suppose $g_0(\boldsymbol{\gamma})$ is polyhedral. Let $\overline{\mu}_j^{\texttt{OLAC}}$ and $\overline{A}_j^{\texttt{OLAC}}$ be the average service rate and average arrival rate to queue j under* OLAC. *Then, if $\theta_j > \tilde{D}_p + \delta_{\max}$, we have:*

$$\overline{\mu}_j^{\texttt{OLAC}} - \overline{A}_j^{\texttt{OLAC}} \leq \delta_{\max} c_p e^{-K_p(\theta_j - \tilde{D}_p - \delta_{\max})}, \ w.p.1. \ \diamond \quad (38)$$

Proof. Omitted due to space limitation. See [24] for details. $\square$

We are now ready to prove Theorem 2.

Proof. (Theorem 2) We define a Lyapunov function $L(t) = \frac{1}{2}\sum_j q_j^2(t)$ and the one-slot conditional drift $\Delta(t) \triangleq \mathbb{E}\big\{L(t+1) - L(t) \mid \boldsymbol{q}(t)\big\}$. Using the queueing dynamic equations (1), we have:

$$\Delta(t) \leq B - \sum_j q_j(t)\mathbb{E}\big\{\mu_j(t) - A_j(t) \mid \boldsymbol{q}(t)\big\}. \quad (39)$$

By adding to both sides the term $V\mathbb{E}\big\{f(t) \mid \boldsymbol{q}(t)\big\} - \sum_j \mathbb{E}\big\{(\beta_j(t) - \theta_j)[\mu_j(t) - A_j(t)] \mid \boldsymbol{q}(t)\big\}$, we obtain:

$$\begin{aligned} &\Delta(t) + V\mathbb{E}\big\{f(t) \mid \boldsymbol{q}(t)\big\} \quad (40)\\ &\quad - \sum_j \mathbb{E}\big\{(\beta_j(t) - \theta_j)[\mu_j(t) - A_j(t)] \mid \boldsymbol{q}(t)\big\} \\ &\leq B + V\mathbb{E}\big\{f(t) \mid \boldsymbol{q}(t)\big\} \\ &\quad - \sum_j \mathbb{E}\big\{(q_j(t) + \beta_j(t) - \theta_j)[\mu_j(t) - A_j(t)] \mid \boldsymbol{q}(t)\big\}. \end{aligned}$$

Using Assumption 2 and plugging in (40) the optimal stationary and randomized policy $\{x_k^{(s_i)*}, \vartheta_k^{(s_i)*}\}_{s_i,k}$, we obtain:

$$\begin{aligned} &\Delta(t) + V\mathbb{E}\big\{f^{\texttt{OLAC}}(t) \mid \boldsymbol{q}(t)\big\} \quad (41)\\ &\quad - \sum_j \mathbb{E}\big\{(\beta_j(t) - \theta_j)[\mu_j^{\texttt{OLAC}}(t) - A_j^{\texttt{OLAC}}(t)] \mid \boldsymbol{q}(t)\big\} \\ &\quad \leq B + V f_{\text{av}}^*. \end{aligned}$$

Here $f^{\texttt{OLAC}}(t)$, $\mu_j^{\texttt{OLAC}}(t)$ and $A_j^{\texttt{OLAC}}(t)$ denote the cost, service rate and arrival rate under the OLAC algorithm. Rearranging the terms, we have:

$$\begin{aligned} &\Delta(t) + V\mathbb{E}\big\{f^{\texttt{OLAC}}(t) \mid \boldsymbol{q}(t)\big\} \leq B + V f_{\text{av}}^* \quad (42)\\ &\quad + \sum_j \mathbb{E}\big\{(\beta_j(t) - \theta_j)[\mu_j^{\texttt{OLAC}}(t) - A_j^{\texttt{OLAC}}(t)] \mid \boldsymbol{q}(t)\big\}. \end{aligned}$$

Taking expectations on both sides of (42) over $\boldsymbol{q}(t)$, taking a telescoping sum over $t = 0, ..., T-1$, rearranging the terms, dividing both sides by TV, and taking a lim sup as $T \to \infty$, we have:

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big\{f^{\texttt{OLAC}}(t)\big\} \leq \frac{B}{V} + f_{\text{av}}^* \quad (43)$$

$$+ \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big\{\sum_j \frac{(\beta_j(t) - \theta_j)}{V}[\mu_j^{\texttt{OLAC}}(t) - A_j^{\texttt{OLAC}}(t)]\big\}.$$

It remains to show that the last term is $O(1/V)$. From Lemma 2 we know that $\beta_j(t) - \theta_j$ converges to $\gamma_j^* - \theta_j = \Theta(V)$ with probability 1. Thus, $w.p.1$, there exists a finite time T_ϵ, such that for all $t \geq T_\epsilon$,

$$|(\beta_j(t) - \theta_j) - (\gamma_j^* - \theta_j)| \leq \epsilon, \ \forall j. \quad (44)$$

Hence, we have:

$$\begin{aligned} &\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big\{\frac{(\beta_j(t) - \theta_j)}{V}[\mu_j^{\texttt{OLAC}}(t) - A_j^{\texttt{OLAC}}(t)]\big\} \\ &\overset{(a)}{\leq} (\frac{\gamma_j^* - \theta_j + \epsilon}{V}) \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big\{\mu_j^{\texttt{OLAC}}(t)\big\} \\ &\qquad - (\frac{\gamma_j^* - \theta_j - \epsilon}{V}) \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big\{A_j^{\texttt{OLAC}}(t)\big\} \\ &\leq \frac{\gamma_j^* - \theta_j}{V}(\overline{\mu}_j^{\texttt{OLAC}} - \overline{A}_j^{\texttt{OLAC}}) + \frac{2\epsilon\delta_{\max}}{V}. \end{aligned} \quad (45)$$

Here (a) follows from the fact that $T_\epsilon < \infty$. Using Lemma 4 with $\theta_j = [\log(V)]^2$ and a sufficiently large V such that $K_p([\log(V)]^2 - \tilde{D}_p - \delta_{\max}) \geq 2\log(V)$, we have:

$$\overline{\mu}_j^{\texttt{OLAC}} - \overline{A}_j^{\texttt{OLAC}} \leq \frac{\delta_{\max} c_p e^{K_p(\tilde{D}_p + \delta_{\max})}}{e^{K_p[\log(V)]^2}} = O(\frac{1}{V^2}). \quad (46)$$

We first consider when $\gamma_{0j}^* > 0$. In this case, $\gamma_j^* = V\gamma_{0j}^* = \Theta(V)$. Thus, (46) implies that:

$$(\gamma_j^* - \theta_j)(\overline{\mu}_j^{\texttt{OLAC}} - \overline{A}_j^{\texttt{OLAC}}) = O(\frac{1}{V}). \quad (47)$$

Using (45) and (46), we conclude that, with probability 1,

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big\{f^{\texttt{OLAC}}(t)\big\} \leq f_{\text{av}}^* + \frac{B}{V} + O(\frac{1}{V}). \quad (48)$$

In the case when $\gamma_j^* = 0$, we see that (47) still holds. Hence, (48) again follows from (47). This completes the proof of Theorem 2. $\square$

Appendix D – Proof of Theorem 4

To prove the theorem, we make use of the following technical lemma from [19].

Lemma 5. *Let $\mathcal{F}_n$ be filtration, i.e., a sequence of increasing σ-algebras with $\mathcal{F}_n \subset \mathcal{F}_{n+1}$. Suppose the sequence of random variables $\{y_n\}_{n\geq 0}$ satisfy:*

$$\mathbb{E}\big\{||y_{n+1} - y^*|| \mid \mathcal{F}_n\big\} \leq \mathbb{E}\big\{||y_n - y^*|| \mid \mathcal{F}_n\big\} - u_n, \quad (49)$$

where u_n takes the following values:

$$u_n = \begin{cases} u & \text{if } ||y_n - y^*|| \geq D, \\ 0 & \text{else.} \end{cases} \quad (50)$$

Here $u > 0$ is a given constant. Then, by defining $N_D \triangleq \inf\{k \mid \|y_n - y^*\| \leq D\}$, we have:

$$\mathbb{E}\big\{N_D\big\} \leq ||y_0 - y^*||/u. \ \diamond \quad (51)$$

Proof. See [19]. $\square$

PROOF. (Theorem 4) From Lemma 3, we see that (26) holds for all t under Backpressure with $\eta < \rho$ and $D_p = \frac{B-\eta^2}{2(\rho-\eta)}$. Hence, using Lemma 5, we have:

$$\mathbb{E}\{T_{D_p}\} \leq ||\boldsymbol{q}(0) - \boldsymbol{\gamma}^*||/\eta.$$

On the other hand, since $\|\boldsymbol{q}(t+1) - \boldsymbol{q}(t)\| \leq B$, i.e., in every time the queue vector can change by at most B distance, we must have $\mathbb{E}\{T_{D_p}\} \geq (||\boldsymbol{q}(0) - \boldsymbol{\gamma}^*|| - D_p)^+/B$. $\square$

Appendix E – Proof of Theorem 5

We prove Theorem 5 in this section. We first have the following lemma regarding the distance between $\boldsymbol{\beta}(t)$ and the distribution estimation accuracy.

Lemma 6. *With probability* 1*, there exists an* $\Theta(1)$ *time* T_{ϵ_s} *(defined in Lemma 1) such that, for all* $t \geq T_{\epsilon_s}$*,*

$$\|\boldsymbol{\beta}(t) - \boldsymbol{\gamma}^*\| \leq 2 \max_{s_i} |\delta_{s_i}(t)| M(V f_{\max} + r\xi B)/\rho. \quad (52)$$

PROOF. See Appendix F. $\square$

Lemma 6 shows that under the polyhedral condition, the distance between the current estimate of the Lagrange multiplier value and the true value diminishes as the empirical distribution converges. This is an important result, as it allows us to focus mainly on the convergence of the distribution when studying the algorithm's convergence. To show our results, we also make use of the following theorem regarding distribution convergence [28].

Theorem 6. *Let* X_1*, ...,* X_n *be independent random variables with* $Pr\{X_i = 1\} = p_i$*, and* $Pr\{X_i = 0\} = 1 - p_i$*. Consider* $X = \sum_{i=1}^n X_i$ *with expectation* $\mathbb{E}\{X\} = \sum_{i=1}^n p_i$*. Then, we have:*

$$Pr\{X \leq \mathbb{E}\{X\} - m\} \leq e^{\frac{-m^2}{2\mathbb{E}\{X\}}}, \quad (53)$$

$$Pr\{X \geq \mathbb{E}\{X\} + m\} \leq e^{\frac{-m^2}{2(\mathbb{E}\{X\}+m/3)}}. \quad \diamond \quad (54)$$

We now state the proof of the convergence time for `OLAC2`.

PROOF. (Theorem 5) Choosing $n = T_l = V^c$ and $m = 4V^{c/2}\log(V)$ in Theorem 6, and let X_t be the indicator of state s_i at time t for $0 \leq t \leq n-1$. Then, we have $\mathbb{E}\{X\} = V^c \pi_{s_i}$. Thus, using the fact that $\pi_{s_i} \leq 1$, we see that at time $t = T_l$,

$$\Pr\{\pi_{s_i}(t) \leq \pi_{s_i} - \frac{4\log(V)}{V^{c/2}}\} \leq e^{-8[\log(V)]^2},$$

$$\Pr\{\pi_{s_i}(t) \geq \pi_{s_i} + \frac{4\log(V)}{V^{c/2}}\} \leq e^{-\frac{8[\log(V)]^2}{1+\frac{2}{3}\log(V)V^{-c/2}}}.$$

Using the union bound, we see that when V is large enough such that $\frac{2}{3}\log(V)V^{-c/2} \leq 1$,

$$\Pr\{\max_{s_i} |\pi_{s_i}(t) - \pi_{s_i}| \geq \frac{4\log(V)}{V^{c/2}}\} \leq Me^{-4[\log(V)]^2}. \quad (55)$$

Therefore, using Lemma 6, we see that with probability at least $1 - \frac{M}{V^{4\log(V)}}$, at time $t = T_l$,

$$\begin{aligned} \|\tilde{\boldsymbol{\beta}} - \boldsymbol{\gamma}^*\| &\leq \frac{8\log(V)M(Vf_{\max} + r\xi B)}{\rho V^{c/2}} \\ &= \Theta(V^{1-c/2}\log(V)), \end{aligned} \quad (56)$$

which implies that at time $t = T_l$,

$$\|\boldsymbol{q}(T_l) - \boldsymbol{\gamma}^*\| = \Theta(V^{1-c/2}\log(V)). \quad (57)$$

Now note that given the same backlog values, `OLAC2` always uses the same actions as LIFO-Backpressure. Using Lemma 3 with $T_{\epsilon_s} = 0$, we see that at any time $t \geq T_l$, if $\|\boldsymbol{q}(t) - \boldsymbol{\gamma}^*\| \geq D_p$, we have:

$$\mathbb{E}\{\|\boldsymbol{q}(t+1) - \boldsymbol{\gamma}^*\| \mid \boldsymbol{q}(t)\} \leq \|\boldsymbol{q}(t) - \boldsymbol{\gamma}^*\| - \eta. \quad (58)$$

Using Lemma 5, we conclude that with probability at least $1 - \frac{M}{V^{4\log(V)}}$:

$$\begin{aligned} \mathbb{E}\{T_{D_p}\} &= O(V^{1-c/2}\log(V)) + T_l \quad (59) \\ &= O(V^{1-c/2}\log(V) + V^c). \quad (60) \end{aligned}$$

This completes the proof of Theorem 5. $\square$

Appendix F – Proof of Lemma 6

Here we prove Lemma 6.

PROOF. (Lemma 6) As in Lemma 1, we have:

$$g(\boldsymbol{\beta}, t) = \sum_{s_i} [\pi_{s_i} + \delta_{s_i}(t)] g_{s_i}(\boldsymbol{\beta}). \quad (61)$$

Also, for any t, $g(\boldsymbol{\beta}(t), t) \geq g(\boldsymbol{\gamma}^*, t)$. By Lemma 1, we see that $w.p.1$, for all $t \geq T_{\epsilon_s}$,

$$|g(\boldsymbol{\beta}(t), t) - g(\boldsymbol{\beta}(t))| \leq \max_{s_i} |\delta_{s_i}(t)| M(V f_{\max} + r\xi B).$$

This implies that:

$$g(\boldsymbol{\gamma}^*) - g(\boldsymbol{\beta}(t)) \leq 2 \max_{s_i} |\delta_{s_i}(t)| M(V f_{\max} + r\xi B). \quad (62)$$

This is so because if (62) does not hold, then:

$$\begin{aligned} & g(\boldsymbol{\beta}(t), t) - g(\boldsymbol{\gamma}^*, t) \\ \leq\ & g(\boldsymbol{\beta}(t)) + \max_{s_i} |\delta_{s_i}(t)| M(V f_{\max} + r\xi B) \\ & - \big[g(\boldsymbol{\gamma}^*) - \max_{s_i} |\delta_{s_i}(t)| M(V f_{\max} + r\xi B)\big] \\ <\ & 0, \end{aligned}$$

which contradicts with $g(\boldsymbol{\beta}(t), t) \geq g(\boldsymbol{\gamma}^*, t)$. Now with (62) and (23), i.e., $g(\boldsymbol{\gamma}^*) - g(\boldsymbol{\beta}) \geq \rho\|\boldsymbol{\gamma}^* - \boldsymbol{\beta}\|$, we see that $w.p.1$, for all $t \geq T_{\epsilon_s} = \Theta(1)$,

$$\|\boldsymbol{\beta}(t) - \boldsymbol{\gamma}^*\| \leq 2 \max_{s_i} |\delta_{s_i}(t)| M(V f_{\max} + r\xi B)/\rho. \quad (63)$$

This completes the proof of Lemma 6. $\square$