Optimal Routing to Parallel Servers With Unknown Utilities—Multi-Armed Bandit With Queues

Xinzhe Fu and Eytan Modiano, *Fellow, IEEE*

***Abstract*— We consider the optimal routing problem in a discrete-time system with a job dispatcher connected to M parallel servers. At every time slot, the job dispatcher sends the incoming jobs to a server for execution, with each server having a queue that stores the jobs. The arrival process of incoming jobs, and the service processes of the servers are stochastic with unknown and possibly heterogeneous rates. Each server s_m is associated with an underlying utility v_m that is initially unknown. Whenever server s_m completes a job, a utility of v_m is obtained and a noisy observation of v_m is received. The goal is to design a policy that makes routing decisions to maximize the total utility obtained by the end of a finite time horizon T. The performance of policies is measured in terms of regret, which is the additive difference between the expected total utility obtained by the policy and the supremum of the expected total utility over all the policies. The optimal routing problem can be interpreted as a problem of multi-armed bandit with queues where each server is viewed as an arm and the completion of a job is viewed as a pull of an arm. The key distinction between the optimal routing problem and traditional multi-armed bandit problems is in the queueing dynamics at the server, which arises due to the stochastic nature of the arrival and service processes. Our results combine techniques from control of stochastic queueing systems and stochastic multi-armed bandits to provide insights to the design and analysis of policies for the optimal routing problem. We first present analytical bounds that link the regret to the utilization and queue length of servers. Next, we start by assuming that the ordering of the underlying utilities is known and introduce the Priority-K routing policy which makes priority-based routing decisions that send the incoming jobs to the server of the highest underlying utility with queue length no larger than a threshold K. We prove that Priority-K achieves $O(\log T)$-regret with an appropriately chosen K. Next, removing the assumption of known utility ordering, we propose the *Upper-Confidence Priority-K* policy, which essentially combines the Priority-K policy with the ordering based on the upper-confidence bounds of the underlying utilities, and establish that the Upper-Confidence Priority-K policy achieves an instance-dependent $O(\log^3 T)$-regret. Finally, we extend our results to the a generalized version of the optimal routing problem with multiple job dispatchers in a bipartite network. Our theoretical results are also validated by simulations.**

***Index Terms*— Queueing analysis, optimization methods.**

Manuscript received 4 April 2022; revised 24 August 2022 and 21 November 2022; accepted 30 November 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Magnusson. This work was supported in part by the Office of Naval Research (ONR) under Grant N00014-20-1-2119 and in part by NSF under Grant CNS-2148128 and Grant CNS-2148183. *(Corresponding author: Xinzhe Fu.)*

The authors are with the Laboratory for Information and Decision Systems, MIT, Cambridge, MA 02139 USA (e-mail: xinzhe@mit.edu).

Digital Object Identifier 10.1109/TNET.2022.3227136

I. INTRODUCTION

CONSIDER a system consisting of a job dispatcher and parallel servers. Incoming jobs arrive at the job dispatcher and get immediately routed to a server where they get queued up for execution. Such system model captures a wide range of applications in communication networks [1], [2], production lines [3], and web server farms [4], and there has been extensive research on the routing problem under this model. Previous works have proposed and analyzed routing policies that aim at optimizing delays [5], minimizing holding costs [6], or achieving desirable load balancing properties [7], [8].

In this paper, we consider the routing problem for parallel servers from the perspective of optimizing system utility. We study the setting where a certain utility is obtained when a server finishes executing a job and the goal is to design a routing policy that maximizes the total obtained utility. The utility associated with each server is unknown apriori but can be learned through the completion of jobs. Such a utility model can represent server-dependent performance measures, such as quality-of-service [9] and energy consumption [10], which the system operator initially has no access to, but can obtain via feedback after job completions.

Specifically, we consider a discrete-time system with a job dispatcher connected to a set of M parallel servers. Jobs arrive at the dispatcher following a stochastic process with an unknown arrival rate. At each time slot, the dispatcher sends each incoming job to one of the servers for execution. Each server s_m has a queue that buffers incoming jobs. The offered service of server s_m at each time slot, i.e., the number of jobs s_m can complete, follows a stochastic process with an unknown service rate. Each server s_m is associated with an underlying utility v_m, and a utility of v_m is attained when s_m completes a job. The underlying utilities are unknown in advance, but for each job completed at s_m, a noisy observation $\hat{v}_m$ with $\mathbb{E}[\hat{v}_m] = v_m$ is received. Our goal is to design routing policies for this system of parallel servers that seek to maximize the utility obtained by the end of a finite time horizon T. We adopt regret as the performance measure, which is defined as the additive difference between the supremum of expected total utilities over all the policies, including the ones that have access to the underlying utilities and network statistics, and the expected total utility of the proposed policy. An important distinction to be made is that we consider an output-queue system, where the incoming jobs are immediately routed to the servers and cannot be held up at the job dispatcher [15], which is in contrast to input-queue

systems [18], where the queue is associated with the dispatcher instead of the servers and the jobs are sent to servers when the servers become available. The output-queue architecture is more suitable for applications where the job dispatcher is not co-located with the servers [16], since the output-queue architecture can effectively reduce the impact of propagation delay on the total turnaround time of jobs and help maintain high system utilization by buffering incoming jobs [17].

The stochastic multi-armed bandit framework [19], [20] studies a system of M arms with each arm associated with an unknown reward. When an arm is played, we obtain and receive a noisy and unbiased observation of the reward of the arm. Viewing each server as a bandit arm, the underlying utilities of the servers as the rewards of the arms, and the execution of a job as a pull of arm, *our optimal routing problem motivates a new class of multi-armed bandit problems where the arms have queues*. The stochasticity in the arrival and service processes and the presence of queues fundamentally reshape the structure of the problem. First, the queues enable the jobs queued up at servers to be completed in a future time slot. Therefore, the arm-playing (job-completion) process is not made deterministically, once every time slot, as in the multi-armed bandit literature [20], [21], [22], [23], [24], [25], [27], [28], [29], but jointly determined by the routing decisions and the stochastic queueing dynamics at the arms. Second, the utility is only obtained upon the completion of jobs. Hence, the unfinished jobs in the queues at the end of the time horizon T, about which the routing decisions have been made, do not contribute to the cumulative utility. This is in stark contrast with the traditional multi-armed bandit framework where the reward is obtained immediately after making a decision [19], [25], [26].

With the new structure of the optimal routing, the theoretical nature of the problem has also changed as the problem is not only about learning the underlying utilities but also leveraging the stochastic queueing dynamics. In the traditional multi-armed bandit framework, the best static policy knowing the underlying utilities (e.g. pulling the arm or the set of arms with the highest rewards) is often optimal and thus the regret is defined with respect to that policy [20], [26]. However, in the optimal routing problem, the best static policy (or any static policy) is provably sub-optimal since it does not take into consideration the queue backlogs (See Section II-C for details). Moreover, the optimal routing problem has unique dimensions of exploration-exploitation trade-offs. The exploration-exploitation trade-off in the traditional framework would only involve exploring the arms sufficiently through routing enough jobs to each server and converging quickly to the servers with high utilities. In our setting, as often the arrival rate is greater than the service rate of any single server, the incoming jobs need to be served by multiple servers, i.e., one server may not be able to satisfy all of the demand. Therefore, the problem goes beyond learning the underlying utilities and identifying servers with high utilities. It requires making intelligent routing decisions to prevent service opportunities of the high-utility servers from being missed due to their queues being empty, while at the same time avoid overloading the servers since unfinished jobs in the queues at the end of the time horizon do not contribute to the total utility. Finally, we note that the queueing dynamics in our problem is different from a recent string of works called "queueing bandit" [30], [31], [32], in which essentially still one arm is played at each time, while the cumulative reward is measured by the queue length, and the goal is to minimize the queue length.

Our main results are to develop analytical bounds and routing policies for the optimal routing problem that combine the techniques from control of stochastic queueing networks and stochastic multi-armed bandits. Specifically, we first establish an upper bound on the regret in terms of of utilization and queue lengths of the servers with high underlying utilities, which reflects the aforementioned exploration-exploitation trade-offs. Next, we propose a routing policy named Priority-K, that essentially routes the incoming jobs to the server with the highest utility whose queue length is no larger than a pre-specified threshold K. When the ordering of the servers' underlying utility is known, we show that Priority-K enjoys $O(\log T)$-regret. We then introduce the policy Upper-Confidence Priority-K, that combines the Priority-K policy with the utility ordering derived from the upper-confidence bounds of the underlying utilities. The Upper-Confidence Priority-K policy enjoys $O(\log^3 T)$-regret through achieving a desirable exploration-exploitation trade-off. Finally, we extend our results to a more general setting where the jobs are coming from multiple job dispatchers, which form a general bipartite topology with the parallel servers.

Note that the optimal routing problem can be subsumed in the framework of network utility maximization with unknown utility functions recently proposed in [11] and [12]. However, using the methods proposed in [11] and [12] cannot achieve logarithmic regret since those methods seek to converge to the best static policy which has a $\Theta(\sqrt{T})$-gap to the optimal as will be discussed in Section II-C, while the policies we propose achieve logarithmic regret. Furthermore, the optimal routing problem can also be formulated as a Markov Decision Process with unknown parameters, and thus can be considered as a reinforcement learning problem [33], [34], [35]. However, existing results on reinforcement learning cannot handle problems with countably infinite state-space. Also, as they cannot exploit the special structure of the optimal routing problem [33], [34], [35], applying the results in reinforcement learning will lead to sub-optimal policies.

The rest of the paper is organized as follows. In Section II, we present the model and problem formulation of the optimal routing problem. In Section III, we establish preliminary upper bounds on the regret. The Priority-K policy and the Upper-Confidence Priority-K policy are introduced and analyzed in Section IV. We evaluate the empirical performance of our policies via simulations in Section V. We further extend our results to the case with multiple job dispatchers in a general bipartite network in Section VI. Finally, in Section VII, we conclude the paper.

II. Model and Problem Formulation

A. System Model

Consider a server farm that operates in discrete time with one job dispatcher and M parallel servers. At each time slot t, there are $a(t)$ jobs that arrive at the dispatcher. The arrivals $a(t)$'s are independent random variables with unknown mean (arrival rate) $\mathbb{E}[a(t)] = \lambda$. The dispatcher routes each job to a

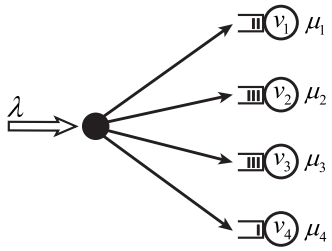


Fig. 1. Illustration of the model.

server. Each server has a queue that stores incoming jobs and employs an arbitrary work-conserving service discipline. For a generic server s_m, its offered service at time t is denoted by $c_m(t)$, with $c_m(t)$'s being independent random variables with unknown mean (service rate) $\mathbb{E}[c_m(t)] = \mu_m$. The offered service $c_m(t)$ is equal to the maximum number of jobs that s_m can finish executing at time t. The arrival rate λ and the service rates $\{\mu_m\}$ will be referred to as the network statistics. After a job j finishes execution at server s_m, we obtain utility v_m and receive a noisy utility observation v_m^j with $v_m^j = v_m + \epsilon_j$ where v_m is the underlying utility associated with server s_m and ϵ_j represents an observation noise that is a 1-sub-Gaussian random variable with $\mathbb{E}[\epsilon_j] = 0$.[1] The noise values ϵ_j's corresponding to different j's are independent. Note that the underlying utility v_m of each server is unknown, and that we can only observe v_m^j's of each job j. We assume that the realized arrivals and offered service rates, i.e., $a(t)$'s, $c_m(t)$'s are all bounded by a constant C almost surely. We also assume $a(t) \geq 1$, $c_m(t) \geq 1$ to avoid unnecessary nuances in the description and analysis of our results.[2] Finally, let $a_m(t)$ be the number of jobs sent to server s_m at time t by the job dispatcher and $Q_m(t)$ be the queue length of server m at t. The evolution of queue length can be written with the Lindley recursion: $Q_m(t+1) := [Q_m(t) + a_m(t) - c_m(t)]^+$, where $[\cdot]^+ = \max\{\cdot, 0\}$. **Figure 1** provides a schematic of our model with four servers.

B. Problem Formulation

We study the problem of designing a routing policy with the maximum utility. Specifically, our goal is to design a policy that makes routing decisions (i.e. sending each incoming job to a server) such that the expected utility obtained by the end of the time horizon T is maximized. To make the problem concrete, we first define the expected utility of a generic policy π. Consider a sample path ω. Suppose on the sample path ω, under the policy π by the end of the time horizon T, server s_m completes $C_m^\pi(\omega, T)$ jobs. Then, the utility obtained under π on the sample path ω is $\sum_{m=1}^M v_m C_m^\pi(\omega, T)$. The expected utility of π over the time horizon is defined as $U_T(\pi) = \mathbb{E}_\omega[\sum_{m=1}^M v_m C_m^\pi(\omega, T)]$. For ease of notation, we will often omit the subscript ω when it is clear that the expectation is taken over all the sample paths. Also, note that only the jobs that are completed by T contribute to the total utility while the jobs that are left in the queue at the end of the time horizon T do not count towards the total utility. Let Π^* be the set of all policies including the ones with knowledge of the underlying utilities and the network statistics or the non-stationary policies. We define the regret of a policy π as $R_T(\pi) = \sup_{\pi^* \in \Pi^*} U_T(\pi^*) - U_T(\pi)$, i.e., the gap in the expected utility between π and supremum over all policies in Π^*. Note that as the set Π^* contains all the policies, the regret essentially characterizes the gap with respect to the best we can hope for. In this paper, we pursue admissible policies that make decisions only based on observable information but not on the network statistics or the underlying utilities. We will refer to the problem as the *Optimal Routing Problem*.

1) Assumptions on the Unknown Statistics: Borrowing the terminology from the multi-armed bandit literature, we focus on the *instance-dependent* regret bound, where there are separations (i.e. constant gaps) between the unknown statistics. Specifically we assume that there do not exists two servers with equal underlying utility, i.e., $\forall m, m'$, $v_m \neq v_{m'}$. Based on this, we will order the servers $s_1, \ldots, s_m$ based on the underlying utility as $v_1 > \ldots > v_M$. Furthermore, we assume that there exists an integer L such that $\sum_{m=1}^L \mu_m < \lambda < \sum_{m=1}^{L+1} \mu_m$, i.e., the arrival rate strictly lies between the total service rates of the first L servers and the first $L+1$ servers. We will refer to L as the *critical number of servers*. The ordering of the servers and the critical number of servers are defined only for the sake of analysis. The policy we propose for the optimal routing problem does not rely on the knowledge of the ordering nor the critical number of servers L. Finally, we define the constant δ as $\min\{\lambda - \sum_{m=1}^L \mu_m, \sum_{m=1}^{L+1} \mu_m - \lambda, 1\}$, which will only be used as an auxiliary constant in the analysis of our policies. Note that for the case of *instance-independent* regret where the aforementioned gaps can scale with the time horizon T or be zero, previous results on multi-armed bandit [20] can easily be extended to the optimal routing problem and show that no algorithm can achieve a regret better than $\Theta(\sqrt{T})$. As such a $\Theta(\sqrt{T})$-regret can already be achieved by existing techniques [12], [13], we do not consider the instance-independent regret in this paper.

We close this section by giving an alternative interpretation of the critical number of servers L. Consider the following optimization problem $\mathcal{P}$.

$$\mathcal{P}: \max_{\{x_m\}} \sum_{m=1}^M v_m x_m \tag{1}$$

$$\textbf{s.t.} \sum_{m=1}^M x_m = \lambda, \tag{2}$$

$$0 \leq x_m \leq \mu_m. \tag{3}$$

$\mathcal{P}$ can be interpreted as a static version of the optimal routing problem, where the optimization variables $\{x_m\}$ can be considered as the steady-state rate of jobs that are sent to each server. Due to the special structure of the optimal routing problem, it is easy to see that the optimal solution $\{x_m^*\}$ to $\mathcal{P}$ is $x_m^* = \mu_m$ for $m = 1, \ldots, L$, $x_{L+1}^* = \lambda - \sum_{m=1}^L \mu_m$, and $x_m^* = 0$ for $m = L+2, \ldots, M$. Therefore, intuitively speaking, a good policy for the optimal routing problem should try to fully utilize the first L servers, send the remaining jobs to the server s_{L+1} and avoid servers $s_{L+2}, \ldots, s_M$.

[1] A random variable X is 1-sub-Gaussian if $\mathbb{P}\{|X| \geq t\} \leq 2\exp(-\frac{t^2}{2})$, i.e., its tail is dominated by a Gaussian distribution with variance 1.

[2] This assumption is to guarantee that there is at least one incoming job at each time slot, and the server can receive at least one utility observation at the time slot if its queue is not empty. Our results still hold without this assumption.

C. Suboptimality of Static Policies

We show that static policies are strictly sub-optimal for the optimal routing problem. The significance of such argument is multi-fold: (i) it highlights the difference between the optimal routing problem, which is a dynamic optimization problem and its static counterpart $\mathcal{P}$, (ii) it also justifies the necessity to define the regret with respect to the supremum over all policies instead of the best static policy that is commonly used in the multi-armed bandit literature [20], [28], [29], (iii) it precludes existing techniques for solving optimization problems with unknown utility functions from achieving superior performance as they focus on constructing policies that converge to the static optimal solution to $\mathcal{P}$. An example is the recent works on network utility optimization with unknown utility functions [11], [12], [13] where methods for solving generalized version of the optimization problem $\mathcal{P}$ (with unknown utility functions and constraint parameters) in network settings were proposed. Finally, it also calls for new techniques for performance analysis and policy design, which we will present in this paper.

For the rest of this section, we will give an intuitive argument that the best static policy based on the optimal solution to $\mathcal{P}$ leads to $\Theta(\sqrt{T})$-regret. In **Appendix A**, we will show more formally a stronger result that there exist instances of the optimal routing problem where any static policy has $\Omega(\sqrt{T})$-regret. As the policy we will propose achieves logarithmic regret, it outperforms any static policy. Note that this includes static policies with access to the network statistics and underlying utilities (as the one following the optimal solution to $\mathcal{P}$), while our policy does not rely on such knowledge.

Consider the following two-server example with $M = 2$, $v_1 > v_2$ and $\lambda = \mu_1 + \frac{\mu_2}{2}$. In this case, it is easy to see that the optimal solution to $\mathcal{P}$ is $x_1^* = \mu_1, x_2^* = \mu_2/2$. The static policy that based on this solution will send a fraction of μ_1/λ of incoming jobs to server s_1 and $\mu_2/(2\lambda)$ of incoming jobs to s_2. Under this policy, the queue of s_1 is critically loaded. It follows that $\mathbb{E}[Q_1(T)] = \Theta(\sqrt{T})$, which means that in expectation, there are $\Theta(\sqrt{T})$ incomplete jobs by the end of the time horizon. If one can divert these jobs to server s_2, the increase in the obtained utility can reach $\Theta(\sqrt{T})$. This is indeed achievable, for example, by the policy we will propose. Hence, the two-server example shows that the static policy based on the optimal solution to $\mathcal{P}$ is bound to have a $\Theta(\sqrt{T})$-regret. Furthermore, it might be tempting to think that a static policy that is a small distance away from the optimal solution to $\mathcal{P}$ could achieve a better regret by avoiding the $\Theta(\sqrt{T})$ queue backlog. We will show in Appendix A that this is not the case, as a static policy that is ϵ-away from the optima of $\mathcal{P}$ would suffer a $\Theta(T\epsilon)$ loss in cumulative utility. The combination of the utility loss and queue backlog would lead to $\Omega(\sqrt{T})$ for any static policy (even when ϵ is taken as a function of T). In contrast, the policy we will propose achieves a superior $O(\log^3 T)$-regret. The sub-optimality of static policies emphasize the need to not only learn the underlying utilities but also leverage the queueing dynamics in the optimal routing problem.

III. Preliminary Results

In this section, we introduce some preliminary bounds on the expected utility and regret that are instrumental in subsequent analysis. These bounds also reflect the exploration-exploitation trade-off faced in the optimal routing problem. We first establish sample path-wise bounds on the total utility of any policy (Propositions 1 and 2), and then use the sample path-wise bound to derive a regret bound (Proposition 3) that links the expected regret to different aspects of the exploration-exploitation trade-off.

On a sample path ω, let $a(t,\omega)$ be the number of jobs that arrived at time t on the sample path, and $c_m(t,\omega)$ be the offered service of server m at time t on the sample path. We first have the following upper bound on the total utility of any policy on the sample path.

Proposition 1:

$$\begin{aligned}&\sup_{\pi^*\in\Pi^*} U_T(\pi^*,\omega)\\ &\quad\le \sum_{m=1}^{L}\sum_{t=0}^{T} c_m(t,\omega)\cdot v_m\\ &\qquad + \left[\sum_{t=0}^{T} a(t,\omega) - \sum_{m=1}^{L}\sum_{t=0}^{T} c_m(t,\omega)\right]\cdot v_{L+1}.\end{aligned}$$

Proof: On the sample path ω, by the end of the time horizon T, each server m can complete at most $\sum_{t=0}^{T} c_m(t,\omega)$ jobs, which would contribute utility of $\sum_{t=0}^{T} c_m(t,\omega)\cdot v_m$. Since $v_1 > \ldots > v_M$, the total utility of any policy on the sample path ω is no larger than the case where a utility of $v_m, m = 1,\ldots,L$ is obtained for $\sum_{t=0}^{T} c_m(t,\omega)$ jobs and a utility of v_{m+1} is obtained for $\sum_{t=0}^{T} a(t,\omega) - \sum_{m=1}^{L}\sum_{t=0}^{T} c_m(t,\omega)$ jobs. Note that the bound still holds for sample paths where $\left[\sum_{t=0}^{T} a(t,\omega) - \sum_{m=1}^{L}\sum_{t=0}^{T} c_m(t,\omega)\right] < 0$. The proposition immediately follows from this bound. □

Note that the bound in Proposition 1 is due to the physical capacity constraints of the servers. Hence, it holds for all the policies in Π^* irrespective of the information that the policies have access to when making decisions. Next, consider an arbitrary policy π. Let $\tilde{c}_m^{\pi}(t,\omega)$ be the realized service of server s_m under policy π on sample path ω. More formally, due to work conservation, $\tilde{c}_m^{\pi}(t,\omega) = \min\{Q_m^{\pi}(t,\omega), c_m(t,\omega)\}$, where $Q_m^{\pi}(t,\omega)$ is the queue length of server s_m under policy π on sample path ω. Then, the following proposition follows from the definition of the total utility.

Proposition 2:

$$U_T(\pi,\omega) = \sum_{m=1}^{M}\sum_{t=0}^{T} \tilde{c}_m^{\pi}(t,\omega)\cdot v_m.$$

From Propositions 1 and 2, we obtain the following upper bound on the regret of a policy on a sample path. Due to space constraints, the proof of Proposition 3 is deferred to **Appendix B-A**.

Proposition 3: For any policy π,

$$\begin{aligned}R_T(\pi) \le &\sum_{t=0}^{T}\sum_{m=1}^{L} \mathbb{P}[Q_m^{\pi}(t) < C]\cdot C^2\\ &+ \mathbb{E}\left[\sum_{m=L+2}^{M}\sum_{t=0}^{T} a_m^{\pi}(t,\omega)\right]\cdot v_{L+1}\\ &+ \mathbb{E}\left[\sum_{m=1}^{L+1} Q_m(T)\right]\cdot v_{L+1}. \qquad (4)\end{aligned}$$

Remark: Proposition 3 essentially captures the aforementioned exploitation-exploration trade-off in the optimal routing problem. As $v_1 > \ldots > v_M$ and $\sum_{m=1}^{L} \mu_m < \lambda < \sum_{m=1}^{L+1} \mu_m$, the first $L+1$ servers correspond to the notion of "high utility servers". In the long run, all the incoming jobs can be served by these $L+1$ servers. Hence, to maximize the total obtained utility, we should try to identify the high utility servers and take full advantage of the service capacity of servers $s_1, \ldots, s_L$ and send the jobs that cannot be served by the top L servers to server s_{L+1}. We should also avoid overloading the servers. The inequality (4) of Proposition 3 gives an analytical bound that exactly captures these trade-offs. The term $\sum_{t=0}^{T} \sum_{m=1}^{L} \mathbb{P}[Q_m^{\pi}(t) < C]$ correspond to the cumulative idleness of the top L servers. The term $\mathbb{E}\left[\sum_{m=L+2}^{M} \sum_{t=0}^{T} a_m^{\pi}(t,\omega)\right]$ represents the total number of jobs sent to the servers with low utilities, which should not be getting excessively many arrivals. Finally, the term $\mathbb{E}\left[\sum_{m=1}^{L+1} Q_m(T)\right]$ represents the total number of unfinished jobs in the high utility servers at the end of the time horizon, which is closely related to the (over)load of the servers.

IV. Routing Policies

In this section, we introduce the policies we propose for the optimal routing problem. We first consider the imaginary case where we know the utility ordering $v_1 > \ldots > v_M$ and propose a prototypical policy for this case. This policy for the imaginary case will be a key building block in the policy we propose for the original routing problem where the utilities are unknown.

A. The Case of Known Utility Ordering

We start with the case where the ordering of the underlying utilities is known. From the preceding discussion, we know that to achieve low regret, we need to reduce the idleness of the top L servers. As the utilities are known, a natural idea is to prioritize the top L servers when making routing decisions for the incoming jobs. However, this idea is not feasible since we do not know the arrival rate and the service rates, and thus have no prior information on the critical number of servers L. In what follows, we show that there exists a policy that achieves logarithmic regret through a utility-based priority routing scheme without the need to know L or the arrival and service rates. We will refer to the policy as the Priority-K policy, with the details shown in **Algorithm 1**.

The Priority-K policy takes a parameter K as input. At each time slot, it inspects the queue length of each server following the decreasing order of utility, i.e., from s_1 to s_M. It sends all the incoming jobs at the current time slot to the first server with queue length no larger than K (Line 6). If all the servers have queue length larger than K, then it sends all the incoming jobs to the last server s_M (Line 8).[3] As its name suggests, priority-K is essentially a priority-based routing policy with a queue length threshold of K, where the priority is determined by the servers' underlying utilities.

[3]As we will show that the probability of all servers have queue length larger than K is in $O(1/T)$, the routing decision in this case can be arbitrary and does not affect the analysis.

Algorithm 1 The Priority-K Policy

Require: Parameter K
1: **Initialize:** $Q_m(0) = 0$ for each m.
2: **for** $t = 0, 1, \ldots,$ **do**
3: $a_m(t) := 0$ for each m.
4: **for** $m = 1, 2, \ldots, M$ **do**
5: **if** $Q_m(t) \leq K$ **then**
6: $a_m(t) := a(t)$.
7: End for loop.
8: **end if**
9: **end for**
10: $a_M(t) := a(t)$ if $Q_m(t) > K$ for all $m = 1, 2, \ldots, M$.
11: Update $Q_m(t)$ for each m.
12: **end for**

The rationale behind Priority-K is that using a priority-based routing policy can effectively reduce the idleness of the servers with high underlying utilities. However, a pure priority-based policy without taking into consideration the service capacities will result in large queue backlogs at the servers at the end of the time horizon, resulting in high regret. Thus, the Priority-K policy uses a threshold K to prevent overloading the servers. Note that the threshold K needs to be set appropriately: if K is too large, then it will fail to serve the purpose of avoiding overloading; if K is too small, it will result in excessive idleness in the servers with high utilities since their queues will frequently become empty as there are insufficient jobs buffered in the queues. In Theorem 1, we will show that Priority-K achieves logarithmic regret when the threshold K is appropriately chosen. Recall that $\delta > 0$ is a constant such that $\delta = \min\{\lambda - \sum_{m=1}^{L} \mu_m, \sum_{m=1}^{L+1} \mu_m - \lambda, 1\}$.

Theorem 1: Let $C_1 = 8CM/\delta$. The regret of the Priority-K policy is in $O(\log T)$, and is achieved with $K = C_1 \log T$.

Proof: Motivated by Proposition 3, we will bound the regret of Priority-K via analyzing $\sum_{t=0}^{T} \mathbb{P}[Q_m^{\pi^{PK}}(t) < C]$ for $m = 1, \ldots, L$ and the terms $\mathbb{E}\left[\sum_{m=L+2}^{M} \sum_{t=0}^{T} a_m^{\pi^{PK}}(t,\omega)\right]$ and $\mathbb{E}\left[\sum_{m=1}^{L+1} Q_m(T)\right]$. For ease of notation, we will omit the superscript π^{PK} in $Q_m^{\pi^{PK}}(t)$, $a_m^{\pi^{PK}}(t,\omega)$ $\tilde{c}_{L+1}^{\pi^{PK}}(t,\omega)$ in the rest of the proof, as the context is clear that they are under the policy π^{PK}. Note that our choice of $C_1 = 8CM/\delta$ requires the knowledge of δ. We will comment at the end on how to choose C_1 without the knowledge of δ.

We begin by giving an outline of the proof that consists of two main steps. The *first step* is to show that $\sum_{t=0}^{T} \mathbb{P}[Q_m(t) \leq C]$ is in $O(\log T)$ for $m = 1, \ldots, L$. The intuition is that considering any $m \leq L$ and the set of servers $\{s_1, \ldots, s_m\}$, as $\lambda > \sum_{i=1}^{m} \mu_i$, the incoming jobs will be sent to the set as long as one of the servers has queue length no larger than K, which means that the arrival rate is greater than the total service rate of $\{s_1, \ldots, s_m\}$. Therefore, if K is large enough, the queue backlog of servers $s_1, \ldots, s_L$ will be maintained at a certain relatively high level and rarely drop below C. The *second step* is to bound the terms $\mathbb{E}\left[\sum_{m=L+2}^{M} \sum_{t=0}^{T} a_m(t,\omega)\right]$ and $\mathbb{E}\left[\sum_{m=1}^{L+1} Q_m(T)\right]$. The term $\mathbb{E}\left[\sum_{m=L+2}^{M} \sum_{t=0}^{T} a_m(t,\omega)\right]$ is the cumulative arrivals

to the servers s_m with $m = L+2, \ldots, M$. Under Priority-K, those servers will only receive incoming jobs when the queue lengths of $s_1, \ldots, s_{L+1}$ are all greater than K. Whereas $\lambda < \sum_{m=1}^{L+1} \mu_m$, i.e., the total capacity of servers $s_1, \ldots, s_{L+1}$ is greater than the arrival rate, so if K is large enough, the event $s_1, \ldots, s_{L+1}$ having queue lengths all greater than K will happen with low probability. The term $\mathbb{E}\left[\sum_{m=1}^{L+1} Q_m(T)\right]$ is the total queue lengths of the top $L+1$ servers at the end of the time horizon, which is deterministically bounded by $O(LK)$ under Priority-K. Note that the three terms we seek to bound project the aforementioned exploration-exploitation trade-off onto choosing the value of K in the context of Priority-K. The first two terms rely on K being large while the third term relies on K being small. Taking K to be in $O(\log T)$ (more specifically, $K = 4CM \log T/\delta$) gives us the desirable trade-off for Priority-K to achieve $O(\log T)$-regret.

We proceed to the details of the proof. From the construction of Priority-K, we have the following observation:

$$\forall m = 1, \ldots, M-1, \ \forall t, \quad Q_m(t) \leq K + C. \tag{5}$$

(5) holds since for $m = 1, \ldots, M-1$, $a_m(t) = 0$ whenever $Q_m(t) > K$, and when $Q_m(t) \leq K$, $a_m(t)$ is bounded by C as the number of incoming jobs $a(t)$ is bounded by C.

The first step: assuming without loss of generality that $L \geq 2$, we start by bounding $\sum_{t=0}^{T} \mathbb{P}[Q_m(t) < C]$ for $m = 1$ (in Lemma 1) and then for $m = 2, \ldots, L$ (in Lemma 2).

Lemma 1: $\sum_{t=0}^{T} \mathbb{P}[Q_m(t) < C] = O(1)$.

Proof of Lemma 1: For server s_1, define $Z_1(t) = K + C - Q_1(t)$. It follows from definition that $\mathbb{P}[Z_1(t) > K] = \mathbb{P}[Q_1(t) < C]$. We further show that $Z_1(t)$ can be interpreted as a potential function with negative one-slot drift when its value becomes larger than C. By definition, we have $Z_1(t) \geq C$ if and only if $Q_1(t) \leq K$, and $|Z_1(t+1) - Z_1(t)| \leq C$ with probability 1. Furthermore, under the priority-K policy, by (5), we have $Z_1(t) \geq 0$ for all t. And when $Z_1(t) \geq C$, which is equivalent to $Q_1(t) \leq K$, we have $a_1(t) = a(t)$. It follows that

$$\begin{aligned}&\mathbb{E}[Z_1(t+1) - Z_1(t) \mid Z_1(t) \geq C] \\ &\quad \leq \mathbb{E}[-a_1(t) + c_1(t) \mid Q_1(t) \leq K] = -(\lambda - \mu_1) < 0.\end{aligned} \tag{6}$$

From the above reasoning, we have that $Z_1(t)$ has negative drift when it exceeds C. Therefore, we should be able to bound the probability of $\{Z_1(t) > K\}$, which is the same as the probability of $\{Q_1(t) < C\}$. In what follows, we make this idea concrete.

From (6), taking $r = \frac{\delta}{4C}$, we can prove the following recursive inequality (7) on $Z_1(t)$. The detailed derivation of (7) is shown in **Appendix B-B**.

$$\mathbb{E}[e^{rZ_1(t+1)}] \leq \left[1 - \frac{(\lambda - \mu_1)r}{2}\right] \cdot \mathbb{E}[e^{rZ_1(t)}] + e^{2rC}. \tag{7}$$

Iterating over inequality (7) and noting that $Z_1(0) = K + C$, we obtain that for any t,

$$\mathbb{E}[e^{rZ_1(t)}] \leq \frac{2e^{2rC}}{(\lambda - \mu_1)r} + \left[1 - \frac{(\lambda - \mu_1)r}{2}\right]^t e^{r(K+C)}. \tag{8}$$

It follows that

$$\mathbb{P}[Z_1(t) > K] = \mathbb{P}[e^{rZ_1(t)} \geq e^{rK}] \leq \frac{\mathbb{E}[e^{rZ_1(t)}]}{e^{rK}} \tag{9}$$

$$\leq \frac{2e^{r(2C-K)}}{(\lambda - \mu_1)r} + \frac{\left[1 - \frac{(\lambda-\mu_1)r}{2}\right]^t e^{r(K+C)}}{e^{rK}}, \tag{10}$$

where we have used Markov's inequality in (9). Plugging in the values $r = \frac{\delta}{4C}$ and $K = C_1 \log T = 4C \log T/\delta$, we have

$$\mathbb{P}[Z_1(t) > K] \leq \frac{8e^{1/2}}{(\lambda - \mu_1)\delta T} + \left[1 - \frac{(\lambda - \mu_1)\delta}{8C}\right]^t e^{\delta/4}. \tag{11}$$

Summing (11) over $t = 0, \ldots, T$,we have $\sum_{t=0}^{T} \mathbb{P}[Q_1(t) < C] = \sum_{t=0}^{T} \mathbb{P}[Z_1(t) > K] = O(1)$. □

Lemma 2: $\sum_{t=0}^{T} \mathbb{P}[Q_m(t) < C] = O(\log T)$ for $m = 2, \ldots, L$.

Proof of Lemma 2: We will only show the details for $m = 2$, as the proof for other $m \leq L$ is essentially the same. For the server s_2, we would need a more sophisticated argument that that for s_1 since the arrival to s_2 depends on the queue length of s_1 while the arrival to s_1 is not affected by the queue length of any other servers under Priority-K. Define $Z_2(t) := 2(K+C) - Q_1(t) - Q_2(t)$. We start by establishing some elementary results on $Z_2(t)$ that show that $Z_2(t)$ can be used to bound $\sum_{t=0}^{T} \mathbb{P}[Q_2(t) < C]$.

First, we show that a bound on $\mathbb{P}[Q_2(t) < C]$ can be obtained by analyzing $\mathbb{P}[Z_2(t) > K]$. Indeed, as $Q_1(t), Q_2(t) \leq K + C$, we again have $Z_2(t) \geq 0$ for all t. Also, as $Q_1(t) \leq K + C$ with probability 1, we have

$$\{Q_2(t) < C\} \subseteq \{Q_1(t) + Q_2(t) \leq K + 2C\} = \{Z_2(t) > K\},$$

which implies that $\mathbb{P}[Q_2(t) < C] \leq \mathbb{P}[Z_2(t) > K]$. Hence, an upper bound on $\mathbb{P}[Z_2(t) > K]$ will lead to an upper bound on $\mathbb{P}[Q_2(t) < C]$.

Second, we establish that $Z_2(t)$ also satisfies a similar drift condition as $Z_1(t)$. Note that $\{Z_2(t) \geq 2C\} = \{Q_1(t) + Q_2(t) \leq 2K\} \subseteq \{Q_1(t) \leq K \text{ or } Q_2(t) \leq K\}$. Therefore, conditioning on $Z_2(t) \geq 2C$, under Priority-K, we have that $a_1(t) = a(t)$ or $a_2(t) = a(t)$ as either server s_1 or server s_2 has queue backlog no larger than K. We thus have $\mathbb{E}[Z_2(t+1) - Z_2(t) \mid Z_2(t) \geq 2C] \leq -(\lambda - \mu_1 - \mu_2) < 0$. Hence following a similar analysis as for $Z_1(t)$, taking $r = \frac{\delta}{8C}$, we have inequality (12) which is proved in **Appendix B-C**.

$$\mathbb{E}[e^{rZ_2(t+1)}] \leq \left[1 - \frac{(\lambda - \mu_1 - \mu_2)r}{2}\right] \cdot \mathbb{E}[e^{rZ_2(t)}] + e^{4rC}. \tag{12}$$

Since $Z_2(0) = 2(K+C)$, we have

$$\begin{aligned}&\mathbb{E}[e^{rZ_2(t)}] \\ &\leq \frac{2e^{4rC}}{(\lambda - \mu_1 - \mu_2)r} + \left[1 - \frac{(\lambda - \mu_1 - \mu_2)r}{2}\right]^t e^{2r(K+C)}.\end{aligned} \tag{13}$$

Similarly as in (10), we have

$$\mathbb{P}[Z_2(t) > K] = \mathbb{P}[e^{rZ_2(t)} \geq e^{rK}] \leq \frac{\mathbb{E}[e^{rZ_2(t)}]}{e^{rK}} \tag{14}$$

$$\begin{aligned}&\leq \frac{2e^{r(4C-K)}}{(\lambda - \mu_1 - \mu_2)r} \\ &\quad + \frac{\left[1 - \frac{(\lambda-\mu_1-\mu_2)r}{2}\right]^t e^{2r(K+C)}}{e^{rK}}.\end{aligned} \tag{15}$$

Again, plugging in the value of r and K, we have that there exists a constant C_2 such that for $t \geq C_2 \log T$,

$\frac{\left[1-\frac{(\lambda-\mu_1-\mu_2)r}{2}\right]^t e^{2r(K+C)}}{e^{rK}} = \left[1-\frac{(\lambda-\mu_1-\mu_2)r}{2}\right]^t e^{r(K+2C)} = O(1/T)$. Combining this with $\frac{2e^{r(4C-K)}}{(\lambda-\mu_1-\mu_2)r} = O(1/T)$, it follows that,

$$\begin{aligned}\sum_{t=0}^{T}\mathbb{P}[Q_2(t)<C] &\le \sum_{t=0}^{T}\mathbb{P}[Z_2(t)>K]\\ &\le \sum_{t=0}^{C_2\log T}\mathbb{P}[Z_2(t)>K]\\ &\quad+\sum_{t=C_2\log T+1}^{T}\mathbb{P}[Z_2(t)>K]\\ &\le \sum_{t=0}^{C_2\log T}\mathbb{P}[Z_2(t)>K]\\ &\quad+O(1)=O(\log T).\end{aligned}$$

Using the same reasoning, we have $\sum_{t=0}^{T}\mathbb{P}[Q_m(t)\le C] = O(\log T)$ for all $m \le L$, where the constructed potential function for server s_m is $Z_m(t) = m(K+C)-\sum_{i=1}^{m}Q_i(t)$. □

The second step: Next, we proceed to the second step and analyze the terms $\mathbb{E}\left[\sum_{m=1}^{L+1}Q_m(T)\right]$ and $\mathbb{E}\left[\sum_{m=L+2}^{M}\sum_{t=0}^{T}a_m(t,\omega)\right]$, For $\mathbb{E}\left[\sum_{m=1}^{L+1}Q_m(T)\right]$, as mentioned in the preceding discussion, by the construction of Priority-K, we have $\mathbb{E}\left[\sum_{m=1}^{L+1}Q_m(T)\right] \le (L+1)(K+C) = O(\log T)$.

For the term $\mathbb{E}\left[\sum_{m=L+2}^{M}\sum_{t=0}^{T}a_m(t,\omega)\right]$, we have the following lemma.

Lemma 3: $\mathbb{E}\left[\sum_{m=L+2}^{M}\sum_{t=0}^{T}a_m(t,\omega)\right] = O(1)$.

Proof of Lemma 3: We start by noting that

$$\begin{aligned}&\mathbb{E}\left[\sum_{m=L+2}^{M}\sum_{t=0}^{T}a_m(t,\omega)\right]\\ &\le \mathbb{E}\left[\sum_{t=0}^{T}\mathbb{1}\{Q_1(t,\omega)>K,\ldots,Q_{L+1}(t,\omega)>K\}\right] \quad (16)\\ &\le \sum_{t=0}^{T}\mathbb{P}\left\{\sum_{m=1}^{L+1}Q_m(t)>(L+1)K\right\}, \quad (17)\end{aligned}$$

where (16) follows from the construction of the Priority-K policy and (17) holds as $\{Q_1(t,\omega)>K,\ldots,Q_{L+1}(t,\omega)>K\}$ implies $\left\{\sum_{m=1}^{L+1}Q_m(t,\omega)>(L+1)K\right\}$.

Consider the function $Z_{L+1}(t) = \sum_{m=1}^{L+1}Q_m(t)$. We again show that $Z_{L+1}(t)$ can serve as a potential function that satisfies a negative drift condition. As $Q_m(t) \le K+C$, we have $\{Z_{L+1}(t) \ge LK+(L+1)C\} \subseteq \{\forall 1 \le m \le L+1, Q_m(t) \ge C\}$. Therefore, under the condition $Z_{L+1}(t) \ge LK+(L+1)C$, the realized services of servers $s_1,\ldots,s_L$ is equal to the offered services. It follows that

$$\begin{aligned}&\mathbb{E}[Z_{L+1}(t+1)-Z_{L+1}(t)\mid Z_{L+1}(t)\ge LK+(L+1)C]\\ &\qquad\le \mathbb{E}[a(t)-\sum_{m=1}^{L+1}c_m(t)] = \lambda-\sum_{m=1}^{L+1}\mu_m<0.\end{aligned} \quad (18)$$

Following a similar argument and taking $r = \frac{\delta_1}{4(L+1)C}$, we establish the following recursive inequality for $Z_{L+1}(t)$ with the derivation deferred to **Appendix B-D**.

$$\begin{aligned}&\mathbb{E}[e^{rZ_{L+1}(t+1)}]\\ &\le \left[1-\left(\sum_{m=1}^{L+1}\mu_m-\lambda\right)r\right]\cdot\mathbb{E}[e^{rZ_{L+1}(t)}]+e^{r(2(L+1)C+LK)}.\end{aligned} \quad (19)$$

As $Z_{L+1}(0)=0$, similarly to the analysis of $Z_1(t)$ and $Z_2(t)$, we have

$$\begin{aligned}\mathbb{P}[Z_{L+1}(t)>(L+1)K] &\le \frac{e^{r(2(L+1)C-K)}}{\left(\sum_{m=1}^{L+1}\mu_m-\lambda\right)r}\\ &\quad+\frac{1}{e^{r(L+1)K}}.\end{aligned} \quad (20)$$

Plugging in the value of r and k, we have that $\sum_{t=0}^{T}\mathbb{P}\{Z_{L+1}(t)>(L+1)K = O(1)$. Therefore, from (17), we have $\mathbb{E}\left[\sum_{m=L+2}^{M}\sum_{t=0}^{T}a_m(t,\omega)\right] = O(1)$. □

Combining the analysis of the two steps and Lemmas 1, 2, and 3, we have $R_T(\pi^{PK}) = O(\log T)$ and conclude the proof of the theorem. From the proof, we can see that the hidden factor in the $O(\log T)$ regret bound is proportional to $\frac{M}{\delta}$ where M is the number of servers and $\delta = \min\{\lambda-\sum_{m=1}^{L}\mu_m, \sum_{m=1}^{L+1}\mu_m-\lambda, 1\} > 0$.

Choice of K: from the preceding analysis we can see that to achieve $O(\log T)$-regret, it suffices to choose $K = C_1\log T$ where C_1 is a constant greater than $4CM/\delta$. Therefore, we do not have to know the exact value but only a lower bound of δ. An alternative way to choose K that does not depend any information on δ is to set $K = \log T\cdot\log\log T$. As T goes to infinity, $\log\log T$ is guaranteed to be larger than any constant independent of T, and going through the same proof, it can be shown that the Priority-K policy with $K = \log T\cdot\log\log T$ achieves $O(\log T\cdot\log\log T)$-regret. In practice, using a good estimation of the lower bound of δ would often lead to a less conservative choice of K and better performance.

Instance-Independent Regret: when δ is 0 or can scale with the time horizon T, we can set K to be $\sqrt{T}\log T$. Following the same analysis, it can be shown that the priority-K policy achieves $\tilde{O}(\sqrt{T})$-regret, which essentially recovers the same regret bound achieved by the optimal static policy in instance-independent cases. □

B. The Case of Unknown Utility Ordering

We now return to the original setting of the optimal routing problem where the ordering of the underlying utilities $v_1,\ldots,v_M$ is unknown. We will propose a routing policy that achieves $O(\log^3 T)$-regret, without relying on the knowledge of the utilities. For the sake of analysis, the servers are still ordered such that $v_1 > \ldots > v_M$. However, our policy will not make use of this ordering.

1) Concentration Inequalities and Upper-Confidence-Bound: We start by introducing some concentration inequalities and the upper-confidence bound that will be used in subsequent analysis. For each server s_m, we define $N_m(t)$ as the cumulative number of utility observations (i.e., cumulative number of jobs completed) at time t, and

$v_m(t)$ as the empirical mean of the utility observations, i.e., $v_m(t) = \frac{1}{N_m(t)} \sum_{j=1}^{N_m(t)} v_m^j$. Since each observation noise ϵ_j is independent and 1-sub-Gaussian, using Chernoff bound [20], we have

$$\mathbb{P}(v_m - v_m(t) \geq \epsilon) \leq \exp\{-N_m(t)\epsilon^2/2\} \quad (21)$$

$$\mathbb{P}(v_m - v_m(t) \leq -\epsilon) \leq \exp\{-N_m(t)\epsilon^2/2\}. \quad (22)$$

Similar to the multi-armed bandit literature [20], we define the upper-confidence-bound of underlying utility v_m based on its empirical mean as $\hat{v}_m(t) := v_m(t) + \sqrt{\frac{4 \ln T}{N_m(t)}}$. We first have the following lemma, that shows that the upper-confidence-bound of each server is greater than its underlying utility with high probability.

Lemma 4: Let E_1 be the event that for all t, m, $\hat{v}_m(t) > v_m$. $\mathbb{P}(E_1) = 1 - O(1/T)$.

Proof: Taking $\epsilon = \sqrt{\frac{4 \ln T}{N_m(t)}}$. For each m, from (21), we have $\mathbb{P}\{\hat{v}_m(t) \leq v_m\} \leq 1/T^2$. Therefore, by the union bound, we have the probability of there exists a time t and server s_m such that $\hat{v}_m(t) \leq v_m$ is in $O(1/T)$, from which the lemma follows. □

Let $\Delta_m = v_{m-1} - v_m$ and $\Delta = \min_m \Delta_m > 0$. Define $\hat{N} = \frac{64 \ln T}{\Delta^2}$. The next lemma shows that the upper-confidence-bound of server s_m will be no larger than the underlying utility of the server s_{m-1} after a sufficient number ($\hat{N}$) of utility observations.

Lemma 5: Let E_2 be the event that $\hat{v}_m(t) < v_{m-1}$ for all t, m such that $N_m(t) \geq \hat{N}$. $\mathbb{P}(E_2) = 1 - O(1/T)$.

Proof: When $N_m(t) \geq \hat{N}$, we have $\sqrt{\frac{4 \ln T}{N_m(t)}} \leq \sqrt{\frac{4 \ln T}{\hat{N}}} \leq \frac{\Delta}{4} \leq \frac{v_{m-1} - v_m}{4}$. It then follows from (22) that when $N_m(t) \geq \hat{N}$, for all m,

$$\begin{aligned} &\mathbb{P}\left\{\hat{v}_m(t) \geq v_m + \frac{v_{m-1} - v_m}{2}\right\} \\ &\quad \leq \mathbb{P}\left\{v_m(t) \geq v_m + \frac{v_{m-1} - v_m}{4}\right\} \\ &\quad \leq \exp\left\{-\frac{N_m(t)}{2}\left(\frac{v_{m-1} - v_m}{4}\right)^2\right\} \leq \frac{1}{T^2}. \end{aligned} \quad (23)$$

Again, using a union bound and noting that $v_{m-1} > v_m + \frac{v_{m-1} - v_m}{2}$ the lemma follows. □

Combining Lemmas 4 and 5, we have Lemma 6.

Lemma 6: Let E_3 be the event that when $N_m(t) \geq \hat{N}$, $\hat{v}_m(t) < v_{m'}$ for all $m' \leq m - 1$. $\mathbb{P}(E_3) = 1 - O(1/T)$.

2) The Upper-Confidence Priority-K Policy: In this section, we present our policy for the optimal routing problem when the utilities are unknown. An easy extension of the Priority-K policy that can achieve logarithmic regret is one that starts with a pure-exploration phased by first sending $\hat{N}$ jobs to each server, and then performs Priority-K for the rest of the time horizon based on the ordering obtained from the utility observations from the first $\hat{N}$ jobs at each server. However, such a policy needs to know the value of $\hat{N}$, or equivalently to know the minimum gap Δ between the underlying utilities. We will propose a policy that dynamically interleaves exploration and exploitation, thereby avoids the need to know $\hat{N}$ or Δ, while still achieves logarithmic regret. We will refer to our policy as the Upper-Confidence Priority-K Policy (UCPK).

UCPK maintains an ordering $\mathcal{O}$ which basically reflects the ordering of the upper-confidence-bounds of the underlying utilities $\{\hat{v}_1(t), \ldots, \hat{v}_M(t)\}$. The details of the UCPK policy are presented in **Algorithm 2**.

Algorithm 2 The Upper-Confidence Priority-K Policy

Require: Parameter K

1: **Initialize:** $\mathcal{O}$ as an arbitrary ordering.
2: **for** $t = 0, \ldots, T$ **do**
3: **if** $\mathcal{O}$ has not changed **then**
4: Allocate servers to incoming jobs using Priority-K based on $\mathcal{O}$.
5: Update $\mathcal{O}$ based on $\{\hat{v}_1(t), \ldots, \hat{v}_M(t)\}$.
6: **else**
7: For the next M time slots send jobs to each server in a round-robin manner, i.e., send the incoming jobs at time slot $t + m - 1$ to server s_m, $m \in \{1, \ldots, M\}$.
8: Update $\mathcal{O}$ based on $\{\hat{v}_1(t+M-1), \ldots, \hat{v}_M(t+M-1)\}$.
9: **end if**
10: **end for**

In UCPK, the ordering $\mathcal{O}$ is initialized to an arbitrary ordering. UCPK has two modes. In the first mode (Lines 4 and 5), when $\mathcal{O}$ has not changed from the update of the upper confidence bounds, then UCPK makes routing decisions using the Priority-K policy (**Algorithm 1**) with the priority $\{1, \ldots, M\}$ replaced by $\mathcal{O}$, i.e., it inspects the queues of the server and sends the incoming jobs to the first server with queue length no larger than K following the ordering $\mathcal{O}$. If the ordering $\mathcal{O}$ has changed, then UCPK goes into the second mode, where it send the incoming jobs at each of the next M time slots to each of the servers $1, \ldots, M$, and update $\mathcal{O}$ based on the upper confidence bounds at the end of the next M time slots. This mode can be considered as pure exploration, where the policy ensures that we obtain at least one utility observation from each server.

We now proceed to analyze the performance of the UCPK policy. We show that it also achieves a logarithmic regret if the parameter K is appropriately chosen.

Theorem 2: The Upper-Confidence-Priority-K policy π^{UCPK} achieves $O(\log^3 T)$-regret with $K = C_1 \log T$, where C_1 is the same constant as in Theorem 1.

Proof: It can be seen from **Algorithm 2** that the execution of UCPK policy can be divided into periods, with each period corresponding to an ordering and a period ends if the ordering $\mathcal{O}$ changes. We define a period as correct if in the corresponding ordering, the first $L + 1$ servers are exactly $s_1, \ldots, s_{L+1}$, i.e., the first $L + 1$ positions in the ordering match the ordering of the underlying utility. Otherwise, the period is incorrect.

To give the main idea of the proof, we identify three factors that contribute to the regret of the UCPK policy. The first comes from the changes in ordering $\mathcal{O}$, as with each change come M time slots of pure exploration which inherently contains sub-optimal decisions. The second comes from the routing decisions made when using Priority-K based on an incorrect ordering. The third comes from the regret accrued by Priority-K with a correct ordering, which has been analyzed in Theorem 1. To bound the regret from the first factor, we need

to bound the total number of periods (changes of $\mathcal{O}$). To bound the regret from the second factor, we need to bound the length of the incorrect periods where the policy follows Priority-K but with a wrong priority. Finally, for the third factor, we can invoke previous results on Priority-K (i.e., from Theorem 1).

Recall from Proposition 1 and Proposition 3 that

$$\begin{aligned} R_T(\pi^{UCPK}) \leq \mathbb{E}\Bigg[&\sum_{t=0}^{T} a(t,\omega) \\ &- \sum_{m=1}^{L}\sum_{t=0}^{T} c_m(t,\omega) - \sum_{t=0}^{T} \tilde{c}_{L+1}(t,\omega)\Bigg] \cdot v_{L+1} \\ &+ \sum_{t=0}^{T}\sum_{m=1}^{L} \mathbb{P}[Q_m(t) < C] \cdot C^2 \\ = \mathbb{E}\Bigg[&a(t,\omega) - \sum_{m=1}^{L} c_m(t,\omega) - \tilde{c}_{L+1}(t,\omega)\Bigg] \\ &\cdot v_{L+1} \\ &+ \sum_{t=0}^{T}\Bigg[\sum_{m=1}^{L} \mathbb{P}[Q_m(t) < C] \cdot C^2\Bigg], \end{aligned}$$

where again, we omit the superscript of $Q_m(t)$ and $\tilde{c}_{L+1}(t,\omega)$ for ease of notations. Define the term $\bar{R}(\pi^{UCPK}, t)$ as

$$\begin{aligned} \bar{R}(\pi^{UCPK}, t) = &\sum_{m=1}^{L} \mathbb{P}[Q_m(t) < C] \cdot C^2 \\ &+ \mathbb{E}\Bigg[a(t,\omega) - \sum_{m=1}^{L} c_m(t,\omega) \\ &- \tilde{c}_{L+1}(t,\omega)] \cdot v_{L+1}. \end{aligned}$$

We define the set of time slots when UCPK is doing pure exploration (Line 4) as $\mathcal{T}_1$, the set of time slots when UCPK is making decisions using Priority-K with an incorrect ordering as $\mathcal{T}_2$, and the time slots when UCPK is making decisions using Priority-K with a correct ordering as $\mathcal{T}_3$.[4] Based on these definitions, we can decompose the regret based on the aforementioned three factors as

$$\begin{aligned} R_T(\pi^{UCPK}) &\leq \sum_{t\in\mathcal{T}_1} \bar{R}(\pi^{UCPK}, t) + \sum_{t\in\mathcal{T}_2} \bar{R}(\pi^{UCPK}, t) \\ &\quad + \sum_{t\in\mathcal{T}_3} \bar{R}(\pi^{UCPK}, t) \qquad (24) \\ &\leq 2MC^2|\mathcal{T}_1| + 2MC^2|\mathcal{T}_2| \\ &\quad + \sum_{t\in\mathcal{T}_3} \bar{R}(\pi^{UCPK}, t), \end{aligned}$$

(25)

where (25) follows from a trivial upper bound $\bar{R}(\pi^{UCPK}, t) \leq 2MC^2$.

We now proceed to bound the three terms on the right-hand-side of (25). Recall that $K = C_1 \log T$ for a sufficiently large constant C_1 which can be chosen the same way as in Theorem 1.

Lemma 7: With probability at least $1 - O(1/T)$, there are at most $\hat{N}$ periods, and $|\mathcal{T}_1| = O(\log T)$.

[4]Rigorously speaking, $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ are random sets that depend on the sample paths. For simplicity of notations, we omit such dependence as it does not affect our proof.

Proof of Lemma 7: Note that we will obtain at least one utility observation from each server at the beginning of each period. From Lemma 6, we have that for any server s_m, if we have obtained at least $\hat{N}$ utility observations of s_m, then there will be at least $m-1$ servers $s_{m'}$ with $\hat{v}_m(t) < \hat{v}_{m'}(t)$. Therefore, if we have obtained at least $\hat{N}$ observations for each server, then for each m, every server with a higher underlying utility than s_m will have a higher upper-confidence-bound than $\hat{v}_m(t)$. Hence, the resulting ordering $\mathcal{O}$ based on the upper-confidence-bound will coincides with the ordering based on the underlying utility (i.e., $s_1, \ldots, s_M$) and will not change further. Thus, with probability $1 - O(1/T)$, there will be at most $\hat{N}$ periods, and it follows that $|\mathcal{T}_1| \leq M\hat{N} = O(\log T)$. □

Lemma 8: With probability at least $1 - O(1/T)$, $|\mathcal{T}_2| \leq O(\log^3 T)$.

Proof of Lemma 8: Let $\mathcal{O}$ be the corresponding ordering of an incorrect period and let $m \leq L+1$ be the first "out-of-place" server in the ordering. More formally, m is the smallest integer where server s_m occupies the j-th position in $\mathcal{O}$ with $j < m$. Such m is well-defined as the period is incorrect.

For this period, first there are M slots with each slot each server gets at least one job. If the ordering changes after the first M slots, then the length of the period is obviously in $O(MK\hat{N})$. Next, we consider the second phase of the period where UCPK follows priority-K with the ordering $\mathcal{O}$. Consider the first m servers in the ordering $\mathcal{O}$. We will show that with probability $1 - O(1/T)$, the length of the period is bounded by $O(\log^2 T)$. Note that the period ends when the ordering $\mathcal{O}$ based on the current upper-confidence bounds changes. And from Lemma 6, we have that the receiving $\hat{N}$ utility observations from server s_m is sufficient for the ordering to change with high probability, since after receiving $\hat{N}$ utility observations, there will be at least $m-1$ servers with upper confidence bounds greater than m, and thus $\hat{v}_m(t)$ will no longer occupies the j-th position ($j < m$) as in the original ordering $\mathcal{O}$ of the period.

We now proceed to bound the length of the time between server s_m receives two utility observations (completes two jobs) under the current ordering $\mathcal{O}$. Let t_0 be any time slot under the current ordering that server s_m completes a job (receives a utility observation). Let $t_0 + \tau$ be the first time slot after t_0 that s_m completes a job. We will show that $\mathbb{P}[\tau = O(\log T)] = O(1/T^2)$. Since the offered service of s_m is lower-bounded by one, whenever the queue of s_m is not empty, it will complete at least one job every time slot. Therefore, $\tau \leq 1$ if $Q_m(t_0+1) > 0$. Furthermore, we have

$$\begin{aligned} &\forall t \in \{t_0+1, \ldots, t_0+\tau-1\}, \\ &Q_m(t) = 0, \text{and } \exists i \in \{1, \ldots, j-1\}, \\ &Q_i(t) \leq K. \end{aligned} \qquad (26)$$

Consider a new system consisting of only servers $s_1, \ldots, s_{j-1}$ with queue lengths $Q_1(t_0), \ldots, Q_{j-1}(t_0)$ being the same as those of the original system. Observe that from t_0 to $t_0+\tau-1$, the state of queue lengths of $s_1, \ldots, s_{j-1}$ evolves identically as that of the new system under UCPK. Furthermore, let $Z(t) = (j-1)(K+C) - \sum_{i=1}^{j-1} Q_i(t)$. In the new system, we have for some $i \in \{1, \ldots, j-1\}$, $a_i(t) = a(t)$, that is, the incoming jobs at time t are routed to one of the

servers $s_1, \ldots, s_{j-1}$. It follows that $\mathbb{E}[Z(t+1) - Z(t) \mid Z(t)] = \mathbb{E}[-a(t) + \sum_{i=1}^{j-1} c_i(t) \mid Z(t)] = -\lambda + \sum_{i=1}^{j-1} \mu_i < 0$. It follows that $Z(t)$ is a super-martingale with one-slot negative drift. As $Z(t) \leq (j-1)(K+C) = O(\log T)$ with probability 1, from the hitting time bound on super-martingales with negative drift in [36] (Theorem 2.3 therein), we have that starting from state, the hitting time of the set $\{Z(t) < C\}$ is in $O(\log T)$ with probability at least $1 - O(1/T^2)$. More formally, starting from any $Z(t_0)$, there exists a $\tau_0 = O(\log T)$ such that with probability at least $1 - O(1/T^2)$, there exists $t \in \{t_0 + 1, \ldots, t_0 + \tau_0\}$ such that $Z(t) \leq C$. Note that as $Q_i(t) \leq K + C$ with probability 1, $\{Z(t) < C\}$ implies that for all $i \in \{1, \ldots, j-1\}$, $Q_i(t) > K$. It follows that in the original system, the interval $\tau = O(\log T)$ with probabilities at least $1 - O(1/T^2)$, i.e., (26) can hold for at most $O(\log T)$ time slots with probability $1 - O(1/T^2)$. Applying a union bound, it follows that with probability at least $1 - O(1/T)$, the time between server s_m receives any two utility observations is in $O(\log T)$. Combining with the preceding discussion, the length of the current incorrect period is upper bounded by the time it takes for s_m to receive $\hat{N}$ utility observations. Thus, we have proved that the length of an incorrect period is bounded by $O(\log^2 T)$. As the total number of periods is in $O(\log T)$, it follows that $|\mathcal{T}_2| \leq O(\log^3 T)$. □

Now, back to the proof of the theorem, for the third term, note that we have already analyzed the term $\bar{R}(\pi^{UCPK}, t)$ in Theorem 1. The analysis there was done for a specific initial state with $Q_m(0) = 0$ for all m. But it is straightforward to see from the analysis that the bound hold for arbitrary initial state with $Q_m(0) \geq 0$. Therefore, we have from Theorem 1 that for each correct period, the sum of $\bar{R}(\pi^{UCPK}, t)$ is in $O(\log T)$. It follows from Lemma 7 that $\sum_{t \in \mathcal{T}_3} \bar{R}(\pi^{UCPK}, t) = O(\log^2 T)$. In summary, combining this with Lemmas 7 and 8, we have that $R_T(\pi^{UCPK}) = O(\log^3 T)$ and conclude the proof. □

Remark: (i). The UCPK policy achieves $O(\log^3 T)$-regret, which strictly dominates all static polices as they have a regret of $\Omega(\sqrt{T})$. The key to this improvement is that UCPK is queue-length-aware, i.e., it takes the queue lengths into consideration when making routing decisions. Such queue-length awareness makes the routing policy dynamic and adaptive to the realizations of the arrival and offered services. (ii). It is possible to show that the UCPK policy still achieves poly-logarithmic regret without the "pure-exploration" slots in $\mathcal{T}_1$ (i.e. Line 4 of Algorithm 2). However, the proof would be much more involved, as we would need to bound the total number of incorrect periods through arguing that for each incorrect period, an "out-of-place" server will receive $\hat{N}$ jobs eventually if the ordering does not change in the meantime. We choose to present the version of UCPK with the "pure-exploration" slots as the analysis is much cleaner. (iii). The instance-dependent factor in the $O(\log^3 T)$-regret of UCPK is proportional to $\frac{M}{\delta \cdot \Delta^2}$, where $\Delta = \min_m \Delta_m > 0$ with $\Delta_m = v_{m-1} - v_m$. (iv). Both the PK and the UCPK policies rely on the knowledge of the time horizon T. We can use the "doubling trick" when the time horizon is unknown: we maintain a pseudo time-horizon T'. Starting from $T' = 1$, we run the policy as if the time horizon is T' (setting K as $C \log T'$). If we have reached the end of the pseudo-time horizon but not the real time horizon, then we double the pseudo-time horizon T' and re-run the policy. Such procedure is continued until the end of the real time horizon. As there are at most $O(\log T)$ pseudo-time horizons, the doubling trick would at most add an $O(\log T)$ factor to the regret.[5] (v). The regret lower bound of $\Omega(\log T)$ for the multi-armed bandit problem [20] also holds for the optimal routing problem since optimal routing is a generalization of the multi-armed bandit problem. However, there is a gap between the $O(\log^3 T)$-regret achieved by the UCPK policy and the lower bound of $\Omega(\log T)$. We conjecture that the stochastic queueing dynamics make the optimal routing problem fundamentally harder than the multi-armed bandit problem, and therefore a lower bound larger than $\Omega(\log T)$ should hold for the optimal routing problem. Proving such a lower bound would narrow or close the gap, which we will leave as future work.

V. Simulations

In this section, we conduct simulations to evaluate the empirical performance of the PK and UCPK policies.

A. Simulation Setup

We consider a network with 20 servers. The underlying utilities of the servers are $\{20, 19, \ldots, 1\}$. The arrival rate to the job dispatcher is 100, and the service rates of the servers are randomly selected from $\{6, 8, 10\}$. We compare the performance of three policies: the best static policy (Static), Priority-K (PK) and Upper-Confidence Priority-K (UCPK). The best static policy makes routing decisions based on the optimal solution to the problem $\mathcal{P}$.

B. Simulation Results

We vary the time horizon T in $\{10000, 20000, \ldots, 200000\}$. For each time horizon, we run the policies and study the queue length at the end of the time horizon (i.e., total number of unfinished jobs), and the utility regret. We compute an upper bound on the regret by using the utility achieved in an imaginary system with one time slot, where the arrival is $\sum_{t=0}^{T} a(t)$ and the offered service of server m is $\sum_{t=0}^{T} c_m(t)$. The parameter K of the PK and UCPK policies is set to $\lceil 20 \log_2 T \rceil$.

We plot the utility regret in Figure 2(a) and the queue length at the end of the time horizon in Figure 2(b). We can see that both the UCPK policy and the PK policy achieve significantly better regret than the best static policy, with the UCPK policy performing slightly worse than the PK policy, due to its need to learn the underlying utilities of the servers. The superiority in terms of utility regret is also manifested in the queue length at the end of the time horizon. As explained in Section II-C, the main drawback of the best static policy is that its queue length can grow as $\Theta(\sqrt{T})$ while for the PK and the UCPK policies, the queue length only grows logarithmically with T.

VI. Extension to General Bipartite Network

In this section, we extend our results to general bipartite networks with multiple job dispatchers. We first present the model for bipartite networks, and then discuss how our proposed policies can be applied to this general setting.

[5]However, since both the UCB and the threshold value K depend on the time horizon, it is more challenging to come up with a more natural "anytime policy" that uses functions of the current time t as the parameters of the policy as in the multi-armed bandit problem.

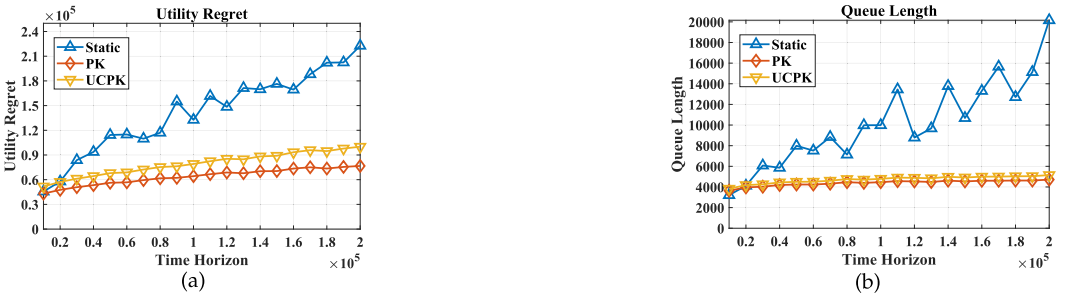


Fig. 2. Utility Regret and Queue Length of Different Policies.

A. Bipartite Network Model

We consider a similar setup as in Section II. Instead of one job dispatcher, we study a system with N job dispatchers and M parallel servers where the dispatcher and servers form a general bipartite graph $\mathcal{G}$. For a dispatcher u_n, we denote by $\mathcal{S}_{u_n}$ the set of servers u_n is connected to. For a server s_m, we denote by $\mathcal{N}_{s_m}$ the set of job dispatchers that have connection to s_m. At each time slot t, there are $a_n(t)$ jobs arriving at dispatcher u_n with $\mathbb{E}[a_n(t)] = \lambda_n$. $a_n(t)$'s are assumed to be i.i.d. and the arrival rates λ_n's are unknown. The service process and the utility observation process are identical to the single dispatcher case in Section II. Note that in this general setting, the utility of a job is still only dependent on the server it is allocated to, but not on the dispatcher it comes from. Let $a_{mn}(t)$ be the number of jobs sent to server s_m from dispatcher u_n at time t ($a_{mn}(t) = 0$ if $s_m \notin \mathcal{S}_{u_n}$). The evolution of the queue length can be written as $Q_m(t+1) := \left[Q_m(t) + \sum_{n=1}^{N} a_{mn}(t) - c_m(t)\right]^+$. Our goal is to design a policy that makes routing decisions for the incoming jobs from each dispatcher. We will refer to the problem as the *generalized optimal routing problem*. Instead of pursuing a policy that achieves logarithmic regret for all instances of the problem, our focus is on exploring whether, when, and how our previous analysis and policies can be extended to this generalized case.

B. Bounds on Utility and Regret

As in Section III, we start by proposing bounds on the utility and regret for the optimal routing problem in general bipartite networks. Recall that the bounds (Propositions 1, 2 and 3) for the optimal routing problem with a single dispatcher was motivated by the solution to the optimization problem $\mathcal{P}$, which is essentially a static version of the optimal routing problem. For the generalized optimal routing problem, we can define a counterpart of $\mathcal{P}$, the optimization problem P', as follows.

$$\mathcal{P}': \max_{\{x_{mn}\}} \sum_{n=1}^{N} \sum_{s_m \in \mathcal{S}_{u_n}} v_m x_{mn} \tag{27}$$

$$\textbf{s.t.} \sum_{m \in \mathcal{S}_{u_n}} x_{mn} = \lambda_n, \tag{28}$$

$$\sum_{n: s_m \in \mathcal{S}_{u_n}} x_{mn} \leq \mu_m. \tag{29}$$

$$0 \leq x_m \leq \mu_m. \tag{30}$$

In $\mathcal{P}'$, the optimization variable x_{mn} can be interpreted as the rate of jobs that dispatcher n sends to server s_m. Let $\{x^*_{mn}\}$ be an optimal solution to $\mathcal{P}'$. We define a server s_m to be *critical* if $\sum_{n: m \in \mathcal{S}_{u_n}} x_{mn} = \mu_m$, a server to be *slack* if $0 < \sum_{n: m \in \mathcal{S}_{u_n}} x_{mn} < \mu_m$, and a server to be *idle* if $\sum_{n: m \in \mathcal{S}_{u_n}} x_{mn} = 0$. Let the set of critical servers be $\mathcal{S}_1$ and the set of slack servers be $\mathcal{S}_2$. For example, in the case with a single job dispatcher, the servers $s_1, \ldots, s_L$ are critical, the server s_{L+1} is slack, while the remaining servers are idle.

The optimal policy for the single job dispatcher case tries to send most of the jobs to critical servers $(s_1, \ldots, s_L)$ to keep them fully utilized, send the remaining jobs to slack servers (in this case there is only one slack server s_{L+1}), and avoid sending jobs to the idle servers. This structure is in line with the optimal solution to $\mathcal{P}$. For a general instance of bipartite networks, the optimal solution to $\mathcal{P}'$ may not have a similar structure. This is mainly because the presence of multiple job dispatchers necessitates global coordination among the dispatchers. As a result, generalized version of the bounds in Section III may not hold and extensions of our proposed policies may not work. This enable us to generalize our previous bounds on utility and regret, and the proposed policies in a non-trivial fashion. We remark here that the conditions are sufficient for our previous analysis and results to be carried over without trivializing the problem. However, they are not necessary for our bounds to hold and we conjecture that even when the conditions are not satisfied, there exist policies with logarithmic regret for the generalized optimal routing problem. However, such policies may be of different nature and require drastically different analysis. We leave the development of such policies to future work.

Recall the bipartite graph $\mathcal{G}$ formed by the job dispatchers and servers. We define the sub-graph $\mathcal{G}_{x^*}$ as the sub-graph of $\mathcal{G}$ with (un-directed) edges such that $x^*_{mn} > 0$. For each slack server $s_m \in \mathcal{S}_2$, we define $\mathcal{G}_m$ as the connected component of $\mathcal{G}_{x^*}$ that s_m belongs to. Furthermore, we define $\tilde{\mathcal{N}}_m$ as the set of job dispatchers in $\mathcal{G}_m$, and $\tilde{\mathcal{S}}_m$ as the set of servers that are in $\mathcal{G}_m$. Now, we are ready to state the two conditions.

Condition 1: For any servers $s_m, s_{m'}$ with $m < m'$, $\mathcal{N}_{s_m} \subseteq \mathcal{N}_{s_{m'}}$ or $\mathcal{N}_{s_{m'}} \cap \mathcal{N}_{s_m} = \emptyset$.

Condition 2: For each $s_m \in \mathcal{S}_1$, there exists a slack server $s_{m'} \in \mathcal{S}_2$ such that $s_m \in \tilde{\mathcal{S}}_{m'}$.

Condition 1 essentially says that for two servers $s_m, s_{m'}$, with s_m having a higher underlying utility than $s_{m'}$, the set of job dispatchers that are connected to s_m is either a subset of or disjoint with that to $s_{m'}$. Condition 2 is the counterpart of the instance-dependent condition on the λ and $\{\mu_m\}$

(i.e., $\sum_{m=1}^{L}\mu_m < \lambda < \sum_{m=1}^{L+1}\mu_m$) in Section II-B.1. It essentially says that there does not exists a connected component in $\mathcal{G}_{x^*}$ without a slack server. The results in this section will all be under Conditions 1 and 2.

Based on the conditions, we show some key structural results for the generalized optimal routing problem in the following four lemmas. The proofs of the lemmas are deferred to **Appendix B-E**.

Lemma 9: There does not exist a job dispatcher u_n that is connected to two slack servers $s_m, s_{m'}$ with $v_m > v_{m'}$ such that $u_n \notin \mathcal{G}_m$ but $u_n \in \mathcal{G}_{m'}$.

Lemma 10: $\forall s_m, s_{m'} \in \mathcal{S}_2$ with $m \neq m'$, $\mathcal{G}_m$ and $\mathcal{G}_{m'}$ do not intersect.

Note that from Condition 2, Lemmas 9 and 10, we have that the collection $\{\tilde{\mathcal{N}}_m, s_m \in \mathcal{S}_2\}$ forms a partition of the set of job dispatchers, and the collection $\{\tilde{\mathcal{S}}_m, s_m \in \mathcal{S}_2\}$ forms a partition of $\mathcal{S}_1 \cup \mathcal{S}_2$.

Lemma 11: For any job dispatcher that is connected to a slack server s_m and an idle server $s_{m'}$, $v_m > v_{m'}$.

Lemma 12: For any slack server s_m, all the critical server in $\tilde{\mathcal{S}}_m$ have underlying utility greater than v_m.

Under Conditions 1 and 2, based on Lemmas 9, 10, 11 and 12, we can show that for the generalized optimal routing problem, we should try to fully utilize the critical servers ($\mathcal{S}_1$) and sends the remaining jobs to slack servers ($\mathcal{S}_2$). Furthermore, each slack server $s_m \in \mathcal{S}_2$ receives all the remaining jobs from the job dispatchers from $\tilde{\mathcal{N}}_m$. Note that since $\tilde{\mathcal{N}}_m$ forms a partition, every job dispatcher is connected to one and only one slack server. This structure enable us to show an upper bound on the utility of any policy, which is formally presented in Proposition 4. The proof is deferred to **Appendix B-F**.

Proposition 4: Under Conditions 1 and 2,

$$\sup_{\pi^*\in\Pi^*} U_T(\pi^*,\omega) \leq \sum_{s_m\in\mathcal{S}_1}\sum_{t=0}^{T} c_m(t,\omega)\cdot v_m + \sum_{s_m\in\mathcal{S}_2}\left[\sum_{t=0}^{T}\sum_{u_n\in\tilde{\mathcal{N}}_m} a_n(t,\omega) - \sum_{t=0}^{T}\sum_{s_{m'}\in\tilde{\mathcal{S}}_m\cap\mathcal{S}_1}^{L} c_{m'}(t,\omega)\right]^+ \cdot v_m.$$

From Proposition 4 and following the same analysis as in the case of single job dispatcher, we obtain the following corollary as an upper bound of regret.

Corollary 1: For any policy π,

$$R_T(\pi) \leq \sum_{t=0}^{T}\sum_{s_m\in\mathcal{S}_1}\mathbb{P}[Q_m^{\pi}(t) < C]\cdot C^2 + \sum_{s_m\in\mathcal{S}_2}\mathbb{E}\left[\sum_{s_{m'}\in\tilde{\mathcal{S}}_m\cap\mathcal{S}_1} Q_{m'}^{\pi}(T,\omega) + \sum_{t=0}^{T}\sum_{u_n\in\tilde{\mathcal{N}}_m}\sum_{\substack{s_{m'}\neq\mathcal{S}_1\\ m'\neq m}} a_{m'n}^{\pi}(t,\omega)\right]\cdot v_m.$$

C. Generalized Routing Policy

From Corollary 1, we see that under Conditions 1 and 2, a policy has low regret if under the policy the idleness of critical servers is low, and the critical servers are not overloaded while the jobs not served by the critical servers are sent to the slack servers (not the idle servers). In what follows, we will show that natural extensions of the PK and the UCPK policies achieves the aforementioned objectives and still enjoy logarithmic regret for the generalized optimal routing problem.

Following a similar road map, we start with the case where the ordering of the underlying utilities is known. For this case, we propose an extension of Priority-K policy that is shown in **Algorithm 3**.

Algorithm 3 The Generalized Priority-K Policy

Require: Parameter K
Initialize: $Q_m(0) = 0$ for each m.
for $t = 0, 1, \ldots,$ **do**
  for $n = 1, \ldots, N$ **do**
    $a_{mn}(t) := 0$ for each m.
    for $m \in \mathcal{S}_{u_n}$ following the decreasing order of underlying utilities **do**
      if $Q_m(t) \leq K$ **then**
        $a_{mn}(t) := a_n(t)$.
      end if
    end for
    if $Q_{m'}(t) > K$ for all $m' \in \mathcal{S}_{u_n}$ then set $a_{mn}(t) := a_n(t)$ for an arbitrary $m \in \mathcal{S}_{u_n}$.
    Update $Q_m(t)$ for each m.
  end for
end for

The Generalized Priority-K policy is a natural extension of the Priority-K policy to the general bipartite network setting. Each job dispatcher u_n follows a local version of the Priority-K policy, inspects the servers in $\mathcal{S}_{u_n}$ following the decreasing order of the underlying utilities and sends the incoming jobs of u_n to the first server with queue length no larger than K.

In Theorem 3, we will show that the generalized Priority-K policy also achieves $O(\log T)$-regret under Conditions 1 and 2. The proof follows a similar idea as in Theorem 1. We construct potential function of queues to bound each term on the right-hand-side of Corollary 1. The details of the proof are presented in **Appendix B-G**.

Theorem 3: Let π^{GPK} be the Generalized Priority-K policy with $K = C_3 \log T$ for some sufficiently large constant C_3. $R_T(\pi^{GPK}) = O(\log T)$.

Based on the Generalized Priority-K policy, we can extend it in the same way to a generalized version of UCPK policy to handle the case of unknown utilities, as the extension from Priority-K to UCPK. It is easy to show that all the analysis for the UCPK policy holds for the Generalized UCPK policy, with the bounds on Priority-K replaced by the bounds on Generalized Priority-K. Therefore, the Generalized UCPK policy achieves $O(\log^3 T)$-regret for the general optimal routing problem. Finally, we note that both the Generalized Priority-K policy and the Generalized UCPK policy are amenable to distributed implementation, as they essentially only require

each dispatcher to make routing decision locally based on the shared information on queue lengths. The flip side is that the local nature of decision making may prevent them from achieving logarithmic regret for instances of generalized optimal routing problem where Conditions 1 and 2 do not hold.

VII. Conclusion

In this paper we studied the problem of optimal routing for parallel servers, which is a new multi-armed bandit formulation where the bandit arms have queues. We presented analytical bounds that link the regret of the policy to the utilization and queue length of servers with high utilities and servers with low utilities, thereby characterizing the exploration-exploitation trade-off in the optimal routing problem. We designed routing policies that enjoy regret that grows logarithmically with the time horizon T. The policies we propose are easy to implement, and have natural extensions which can still achieve logarithmic regret for the generalized optimal routing problem with multiple job dispatchers under certain structural conditions. Despite the simplicity and the good theoretical performance guarantees of our policies for the optimal routing problem with single job dispatcher, we have not been able to extend them to handle all instances of the generalized problem with multiple job dispatchers. Designing efficient routing policies with provably low regret for the generalized optimal routing problem is an important future direction.

Appendix A
Regret Lower Bound of Static Policies

In this section, we establish a lower bound on the regret of static policies. We formally define the static policies as ones under which the numbers of jobs $a_m(0), \ldots, a_m(T)$ sent to each server s_m at time $t = 0, \ldots, T$ are independent random variables with the same mean. Note that we only require independence of decisions corresponding to each server across time, but do not ask for independence of $a_1(t), \ldots, a_M(t)$ across different servers for the same t. The lower bound is summarized as follow.

Proposition 5: There exist instances of the optimal routing problem in which any static policy has $\Omega(\sqrt{T})$-regret.

Proof: We consider an instance with two servers ($M = 2$). Server s_1 has an integer service rate μ_1 with its offered service $c_1(t)$ being an integer chosen from $\{\mu_1 - 1, \mu_1 + 1\}$ uniformly at random. Server s_2 has an integer service rate μ_2 with its offered service $c_2(t)$ being an integer chosen from $\{\mu_2 - 1, \mu_2 + 1\}$ uniformly at random. The underlying utilities $v_1 = v_2 + 1$. The arrival process is deterministic, with $a(t) = \lambda$ for each t. Note that we explicitly specify the distributions of the arrival and service processes only for concreteness. It should be clear from the proof that the result holds for a wide range of instances, not restricted to $M = 2$ or the distributions assumed here.

We consider an arbitrary static routing policy π that sends $a_1(t) = \lambda_1, t = 0, \ldots, T$ jobs to s_1 and $a_2(t) = \lambda_2, t = 0, \ldots, T$ jobs to s_2. By Proposition 1, for the optimal policy π^*, on any sample path ω, $U_T(\pi^*, \omega) \leq \sum_{t=0}^{T} c_1(t, \omega) v_1 + \left[\sum_{t=0}^{T} \lambda - \sum_{t=0}^{T} c_1(t, \omega)\right] v_2$. It follows that

$$\begin{aligned} U_T(\pi^*) &\leq \mathbb{E}\left[\sum_{t=0}^{T} c_1(t,\omega) v_1 \right. \\ &\qquad \left. + \left[\sum_{t=0}^{T} \lambda - \sum_{t=0}^{T} c_1(t,\omega)\right] v_2\right] \\ &= \sum_{t=0}^{T} \mu_1 v_1 + \left[\sum_{t=0}^{T} \lambda - \sum_{t=0}^{T} \mu_1\right] v_2. \end{aligned} \tag{31}$$

We will show that the gap between (31) and $U_T(\pi)$ is $\Omega(\sqrt{T})$. Although (31) upper-bounds and may not be equal to $U_T(\pi^*)$, as the UCPK policy achieves logarithmic regret, having a $\Omega(\sqrt{T})$ gap with respect to (31) implies a gap of the same order with respect to UCPK, from which it will follow that any static policy has $\Omega(\sqrt{T})$-regret.

For the static policy π, by definition, we have

$$U_T(\pi) \leq \left[\sum_{t=0}^{T} \lambda_1 - \mathbb{E}[Q_1(T)]\right] v_1 + \sum_{t=0}^{T} (\lambda - \lambda_1) v_2. \tag{32}$$

It follows that the gap between the right-hand-side of (31) and $U_T(\pi)$ is at least

$$\left[\sum_{t=0}^{T} \mu_1 - \sum_{t=0}^{T} \lambda_1 + \mathbb{E}[Q_1(T)]\right] v_1 + \sum_{t=0}^{T} (\mu_1 - \lambda_1) v_2. \tag{33}$$

Recall the evolution of Q_1 as $Q_1(t+1) = [Q_1(t) + a_1(t) - c_1(t)]^+ \geq Q_1(t) + a_1(t) - c_1(t)$. Let $\tilde{Q}_1(T) = \sum_{t=0}^{T} [a_1(t) - c_1(t)]$. It follows that $\mathbb{E}[Q_1(T)] \geq \mathbb{P}[\tilde{Q}_1(T) \geq 0] \cdot \mathbb{E}[\tilde{Q}_1(T) \mid \tilde{Q}_1(T) \geq 0]$. For any t, $a_1(t) - c_1(t)$ takes value in $\{\lambda_1 - \mu_1 - 1, \lambda_1 - \mu_1 + 1\}$ uniformly at random. It has the same distribution as $\lambda_1 - \mu_1 - 1 + 2 \cdot b(t)$ where $b(t)$ is a Bernoulli random variable that takes value 0 or 1 with equal probability.[6] Therefore, $\tilde{Q}_1(T)$ has the same distribution as $\sum_{t=0}^{T} (\lambda_1 - \mu_1) + 2 \cdot B(T+1, 1/2)$, where $B(T+1, 1/2)$ is a binomial random variable with parameters $T+1$, $1/2$. Let $\delta = \lambda_1 - \mu_1$. It follows that $\mathbb{E}[\tilde{Q}_1(T)] = \sum_{t=0}^{T} \delta$ and as $a_1(t)$, $c_1(t)$ are independent across time, the variance of $\tilde{Q}_1(T)$ is $T+1$. As for the binomial random variable $\tilde{Q}_1(T)$, $\mathbb{P}[\tilde{Q}_1(T) \geq 0] \cdot \mathbb{E}[\tilde{Q}_1(T) \mid \tilde{Q}_1(T) \geq 0]$ is at least the same order as the mean plus the standard deviation of $\tilde{Q}_1(T)$. It follows that if $\delta \geq 0$ or $|\delta| = o(1/\sqrt{T})$, we have

$$\begin{aligned} \mathbb{E}[Q_1(T)] &\geq \mathbb{P}[\tilde{Q}_1(T) \geq 0] \cdot \mathbb{E}[\tilde{Q}_1(T) \mid \tilde{Q}_1(T) \geq 0] \\ &\geq \Omega(T\delta + \sqrt{T}) = \Omega(\sqrt{T}). \end{aligned}$$

If $\delta < 0$ and $|\delta| = \Omega(1/\sqrt{T})$, as still $\mathbb{E}[Q_1(T)] \geq 0$, we have

$$(33) \geq -\sum_{t=0}^{T} \delta = T|\delta| = \Omega(\sqrt{T}).$$

Therefore, in either case, the regret of a generic static policy π is $\Omega(\sqrt{T})$, which concludes the proof. □

[6] Note that here the distribution of $a_1(t) - c_1(t)$ does not matter as long as its variance is a positive constant.

APPENDIX B
ADDITIONAL PROOFS

A. Proof of Proposition 3

Proof: From Propositions 1 and 2, we have

$$\begin{aligned}
&\sup_{\pi^*\in\Pi^*} U_T(\pi^*,\omega) - U_T(\pi,\omega)\\
&\quad\le \sum_{t=0}^{T}\sum_{m=1}^{L}[c_m(t,\omega)-\tilde{c}_m^{\pi}(t,\omega)]\cdot C\\
&\qquad+\left[\sum_{t=0}^{T} a(t,\omega) - \sum_{m=1}^{L}\sum_{t=0}^{T} c_m(t,\omega)\right]\cdot v_{L+1}\\
&\qquad-\sum_{m=L+1}^{M}\sum_{t=0}^{T}\tilde{c}_m^{\pi}(t,\omega)\cdot v_m\\
&\quad\le \sum_{t=0}^{T}\sum_{m=1}^{L}[c_m(t,\omega)-\tilde{c}_m^{\pi}(t,\omega)]\cdot C\\
&\qquad+\left[\sum_{t=0}^{T} a(t,\omega) - \sum_{m=1}^{L}\sum_{t=0}^{T} c_m(t,\omega)\right.\\
&\qquad\left.-\sum_{t=0}^{T}\tilde{c}_{L+1}^{\pi}(t,\omega)\right]\cdot v_{L+1}.
\end{aligned}$$

Due to the work-conserving nature of the servers, we have that $\tilde{c}_m^{\pi}(t,\omega) < c_m(t,\omega)$ only if $Q_m^{\pi}(t,\omega) < C$. It follows that $c_m(t,\omega) - \tilde{c}_m^{\pi}(t,\omega) \le \mathbb{1}[Q_m^{\pi}(t,\omega) < C]\cdot C$. Therefore, we have

$$\begin{aligned}
&\sup_{\pi^*\in\Pi^*} U_T(\pi^*,\omega) - U_T(\pi,\omega)\\
&\quad\le \sum_{t=0}^{T}\sum_{m=1}^{L}\mathbb{1}[Q_m^{\pi}(t,\omega)<C]\cdot C^2\\
&\qquad+\left[\sum_{t=0}^{T} a(t,\omega) - \sum_{m=1}^{L}\sum_{t=0}^{T} c_m(t,\omega)\right.\\
&\qquad\left.-\sum_{t=0}^{T}\tilde{c}_{L+1}^{\pi}(t,\omega)\right]\cdot v_{L+1}. \qquad (34)
\end{aligned}$$

Taking expectation over ω for both sides of (34), we have

$$\begin{aligned}
R_T(\pi) &\le \sum_{t=0}^{T}\sum_{m=1}^{L}\mathbb{P}[Q_m^{\pi}(t)<C]\cdot C^2\\
&\quad+\mathbb{E}\left[\sum_{t=0}^{T} a(t,\omega) - \sum_{m=1}^{L}\sum_{t=0}^{T} c_m(t,\omega)\right.\\
&\quad\left.-\sum_{t=0}^{T}\tilde{c}_{L+1}^{\pi}(t,\omega)\right]\cdot v_{L+1}.
\end{aligned}$$

For the second term on the right-hand-side above, since $\tilde{c}_m^{\pi}(t,\omega)\le c_m(t,\omega)$, we have

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=0}^{T} a(t,\omega) - \sum_{m=1}^{L}\sum_{t=0}^{T} c_m(t,\omega) - \sum_{t=0}^{T}\tilde{c}_{L+1}^{\pi}(t,\omega)\right]\\
&\quad\le \mathbb{E}\left[\sum_{t=0}^{T} a(t,\omega) - \sum_{m=1}^{L+1}\sum_{t=0}^{T}\tilde{c}_m^{\pi}(t,\omega)\right]\\
&\quad\le \mathbb{E}\left[\sum_{m=1}^{L+1} Q_m^{\pi}(T)\right] + \mathbb{E}\left[\sum_{m=L+2}^{M}\sum_{t=0}^{T} a_m^{\pi}(t,\omega)\right],
\end{aligned}$$

from which the proposition follows. □

B. Proof of Inequality (7)

Let $\eta_1(t) = Z_1(t+1) - Z_1(t)$ and recall the constant $r = \frac{\delta}{4C}$. Noting that $|\eta_1(t)|\le C$, we first have

$$e^{rZ_1(t+1)} = e^{rZ_1(t)}\cdot e^{r\eta_1(t)} \le e^{rZ_1(t)}\cdot[1 + r\eta(t) + 2r^2\eta^2(t)],$$

where the second inequality is due to that $e^x \le 1+x+2x^2$ for $0\le x\le 1$, and $r\eta_1(t)\le rC = \delta/4 \le 1$. It follows that

$$\begin{aligned}
\mathbb{E}[e^{rZ_1(t+1)} \mid Z_1(t) < C] &\le e^{rZ_1(t)}\cdot e^{rC} \qquad (35)\\
\mathbb{E}[e^{rZ_1(t+1)} \mid Z_1(t) \ge C] &\le e^{rZ(t)}\cdot[1 + r\eta(t) + 2r^2\eta^2(t)]\\
&\le e^{rZ_1(t)}\cdot\left[1 - \frac{\lambda-\mu_1}{2}r\right], \qquad (36)
\end{aligned}$$

where the last inequality follows from $\mathbb{E}[r\eta(t)\mid Z_1(t)\ge C] = -(\lambda-\mu_1)r$, and $2r^2\eta^2(t) \le -\frac{\eta(t)}{2}r$ since $r|\eta(t)|\le rC \le 1/4$. Write $\delta_1 = \frac{(\lambda-\mu_1)}{2}$. It follows that

$$\begin{aligned}
&\mathbb{E}[e^{rZ_1(t+1)}]\\
&= \mathbb{P}[Z_1(t)\ge C]\cdot\mathbb{E}[e^{rZ_1(t+1)}\mid Z_1(t)\ge C]\\
&\quad+\mathbb{P}[Z_1(t)<C]\cdot\mathbb{E}[e^{rZ_1(t+1)}\mid Z_1(t)<C]\\
&\le e^{rC}\cdot\mathbb{E}[e^{rZ_1(t)}\mid Z_1(t)<C]\cdot\mathbb{P}[Z_1(t)<C]\\
&\quad+[1-\delta_1 r]\cdot\mathbb{E}[e^{rZ_1(t)}\mid Z_1(t)\ge C]\cdot\mathbb{P}[Z_1(t)\ge C]\\
&= [1-\delta_1 r]\cdot\mathbb{E}[e^{rZ_1(t)}]\\
&\quad+\left[e^{rC}-(1-\delta_1 r)\right]\cdot\mathbb{E}[e^{rZ_1(t)}\mid Z_1(t)\le C]\\
&\quad\cdot\mathbb{P}[Z_1(t)\le C]\\
&\le [1-\delta_1 r]\cdot\mathbb{E}[e^{rZ_1(t)}] + e^{2rC}.
\end{aligned}$$

C. Proof of Inequality (12)

Define $\eta_2(t) = Z_2(t+1) - Z_2(t)$ and recall that $r = \frac{\delta}{8C}$. Similar as in the proof of (7).Write $\delta_2 = \frac{(\lambda-\mu_1-\mu_2)}{2}$. We have

$$\begin{aligned}
&\mathbb{E}[e^{rZ_2(t+1)}]\\
&= \mathbb{P}[Z_2(t)\ge 2C]\cdot\mathbb{E}[e^{rZ_2(t+1)}\mid Z_2(t)\ge 2C]\\
&\quad+\mathbb{P}[Z_2(t)<2C]\cdot\mathbb{E}[e^{rZ_2(t+1)}\mid Z_2(t)<2C]\\
&\le e^{2rC}\cdot\mathbb{E}[e^{rZ_2(t)}\mid Z_2(t)<2C]\cdot\mathbb{P}[Z_2(t)<2C]\\
&\quad+[1-\delta_2 r]\cdot\mathbb{E}[e^{rZ_2(t)}\mid Z_2(t)\ge 2C]\cdot\mathbb{P}[Z_2(t)\ge 2C]\\
&= \left[e^{2rC}-(1-\delta_2 r)\right]\cdot\mathbb{E}[e^{rZ_2(t)}\mid Z_2(t)<2C]\\
&\quad\cdot\mathbb{P}[Z_2(t)<2C]\\
&\quad+[1-\delta_2 r]\cdot\mathbb{E}[e^{rZ_2(t)}]\\
&\le [1-\delta_2 r]\cdot\mathbb{E}[e^{rZ_2(t)}] + e^{4rC}.
\end{aligned}$$

D. Proof of Inequality (19)

The inequality (19) is established in an entirely analogous way as (7) and (12). Note that $|Z_L(t+1)-Z_L(t)|\le (L+1)C$. Writing $L_1 = (L+1)C + LK$ and $\delta_m = \left(\sum_{m=1}^{L+1}\mu_m - \lambda\right)$,

$$\begin{aligned}
&\mathbb{E}[e^{rZ_{L+1}(t+1)}]\\
&\quad= \mathbb{P}[Z_{L+1}(t)\ge L_1]\cdot\mathbb{E}[e^{rZ_{L+1}(t+1)}\mid Z_{L+1}(t)\ge L_1]
\end{aligned}$$

$$
\begin{aligned}
&+\mathbb{P}[Z_{L+1}(t)<L_1]\cdot\mathbb{E}[e^{rZ_{L+1}(t+1)}\mid Z_{L+1}(t)<L_1]\\
&\le [1-\delta_m r]\cdot\mathbb{E}[e^{rZ_{L+1}(t)}\mid Z_{L+1}(t)<L_1]\\
&\quad\cdot\mathbb{P}[Z_{L+1}(t)<L_1]\\
&\quad+e^{L'rC}\cdot\mathbb{E}[e^{rZ_{L+1}(t)}\mid Z_{L+1}(t)<L_1]\\
&\quad\cdot\mathbb{P}[Z_{L+1}(t)<L_1]\\
&=[1-\delta_m r]\cdot\mathbb{E}[e^{rZ_{L+1}(t)}]+e^{r(2L'C+LK)}.
\end{aligned}
$$

E. Proofs of Lemmas 9, 10, 11, 12

Proof of Lemma 9: We prove the lemma by contradiction. If there exists a job dispatcher u_n with $u_n \notin \mathcal{G}_m$ but $u_n \in \mathcal{G}_{m'}$ and $v_m > v_{m'}$, then $x^*_{mn} = 0$ and $x^*_{m'n} > 0$. Since $s_m, s_{m'} \in \mathcal{S}_2$, we have $\sum_{n:s_m\in\mathcal{S}_{u_n}} x^*_{mn} < \mu_m$ and $\sum_{n:s_{m'}\in\mathcal{S}_{u_n}} x^*_{m'n} < \mu_{m'}$. Without loss of generality we assume $v_m > v'_m$, we can thus increase x^*_{mn} and decrease $x^*_{m'n}$ by a same sufficiently small amount and obtain a feasible solution with larger utility than that of $\{x^*_{mn}\}$, which contradicts the optimality of $\{x^*_{mn}\}$.

Proof of Lemma 10: Again, we prove the lemma by contradiction. If there exist $s_m, s_{m'} \in \mathcal{S}_2, m \neq m'$ with $\mathcal{G}_m \cap \mathcal{G}_{m'} \neq \emptyset$. Due to the bipartite nature of the graph, it follows that there exists a path of even length from s_m to $s_{m'}$ in $\mathcal{G}_{x^*}$. Hence, by the definition of $\mathcal{G}_{x^*}$, there exists a path $s_m, u_{n_1}, s_{m_1}, \ldots, u_{n_k}, s_{n_k}, u_{n_{k+1}}, s_{m'}$ where the value of the component of x^* corresponding to each edge in the path is positive. Without loss of generality, assume $v_m > v_{m'}$. Then, we can "propagate" a sufficiently small positive value of flow from $s_{m'}$ to s_m, i.e., decreasing $x^*_{m'n_{k+1}}$, increasing $x^*_{m_k n_{k+1}}$,..., increasing $x^*_{mn_1}$. After such propagation, the resulting solution will still remain feasible since both s_m and $s_{m'}$ are slack server. However, as $v_m > v_{m'}$, we now arrive at a solution that has higher value of objective function than the original $\{x^*\}$, which again contradicts the optimality of $\{x^*\}$.

Proofs of Lemmas 11 and 12: The lemmas follow from the same argument as the proof of Lemmas 9 and 10.

F. Proof of Proposition 4

Proof: We prove the proposition by inspecting the connected components corresponding to the slack servers following decreasing order of underlying utilities. We first consider the slack server s_{m_1} with the largest underlying utility in $\mathcal{S}_2$. On a sample path ω, the total number of incoming jobs from the job dispatchers in $\tilde{\mathcal{N}}_{m_1}$ is equal to $\sum_{t=0}^{T}\sum_{u_n\in\tilde{\mathcal{N}_{m_1}}} a_n(t,\omega)$. By Lemmas 11 and 12, the maximum utility that can be obtained from these jobs is upper-bounded by the critical servers using up all the offered services and the remaining jobs being served by the slack server s_{m_1}, which lead to a total utility of at most $\sum_{s_{m'}\in\tilde{\mathcal{S}_{m_1}}\cap\mathcal{S}_1}\sum_{t=0}^{T} c_{m'}(t,\omega)\cdot v_{m'} + \big[\sum_{t=0}^{T}\sum_{u_n\in\tilde{\mathcal{N}}_m} a_n(t,\omega) - \sum_{t=0}^{T}\sum_{s_{m'}\in\mathcal{S}_{m_1}\setminus\{s_{m_1}\}} c_{m'}(t,\omega)\big]^+\cdot v_{m_1}$. We then consider the slack server s_{m_2} with the second largest underlying utility in $\mathcal{S}_2$. Note that by Lemma 9, the job dispatchers in $\tilde{\mathcal{N}}_{m_2}$ are not connected to s_{m_1}, so their traffic cannot be diverted to s_{m_1}. Thus, together with s_{m_1}, a corresponding upper bound in this case still holds. By repeating this argument over all the slack servers and their connected components, we have enumerated all the job dispatchers. The proposition thus follows. □

G. Proof of Theorem 3

Proof: The idea of the proof is the same as Theorem 1. We will use Corollary 1 and bound its right-hand-side through analyzing the term $\mathbb{P}[Q_m(t)]$ for each critical server, and the regret term associated with each slack server. Note that for each slack server s_m, by definition $\sum_{u_n\in\tilde{\mathcal{N}}_m} > \sum_{s_{m'}\in\tilde{\mathcal{S}}_m\cap\mathcal{S}_1}\mu_{m'}$. It follows that the difference between $\mathbb{E}\big[[\sum_{u_n\in\mathcal{N}_m} a_n(t,\omega) - \sum_{s_{m'}\in\mathcal{S}_m\cap\mathcal{S}_1}^{L} c_{m'}(t,\omega)]^+ - \tilde{c}^{\pi}_m(t,\omega)\big]$ and $\mathbb{E}\big[\sum_{u_n\in\mathcal{N}_m} a_n(t,\omega) - \sum_{s_{m'}\in\mathcal{S}_m\cap\mathcal{S}_1}^{L} c_{m'}(t,\omega) - \tilde{c}^{\pi}_m(t,\omega)\big]$ is of order $O(1)$, which means that we can essentially analyze the latter term.

The analysis is also based on constructing suitable potential functions, establishing drift arguments for the potential functions, and deriving upper bounds on the relevant term through the drift arguments. To avoid unnecessary repetition, we will only illustrate how to construct the potential functions and establish corresponding drift arguments. We start from s_1 and go over each server following the decreasing order of underlying utility.

For a server s_m, if it is an idle server, then it does not appear in the right-hand-side of Corollary 1, so we can skip it. If $s_m \in \mathcal{S}_1$, then recall the partition based on the connected component of each slack server. Slightly overloading the notations, we define $\mathcal{S}_1^m$ as the set of critical server with larger underlying utility than s_m in the connected component that s_m belongs to. We construct the potential function $Z_m(t) = \sum_{s'_m\in\mathcal{S}_1^m}[K+C-Q_{m'}(t)]$. By essentially the same argument as in the proof of Theorem 1, we have $\mathbb{P}[Q_m(t) < C] \le \mathbb{P}[Z_m(t) > K]$ as $Q_m(t) < C$ implies that $Z_m(t) > K$. Also, note that due to the Condition 1, there is no slack server with larger underlying utility (higher priority in Generalized Priority-K) than s_m. Hence, with Condition 2, we can establish a similar negative conditional one-slot drift for $Z_m(t)$ as in the proof of Theorem 1, and a bound on $\mathbb{P}[Z_m(t) > K]$ follows similarly. Finally, if $s_m \in \mathcal{S}_2$, then we consider the connected component of the slack server s_m. Same as in the proof of Theorem 1, the term $\mathbb{E}\big[\sum_{u_n\in\mathcal{N}_m} a_n(t,\omega) - \sum_{s_{m'}\in\mathcal{S}_m\cap\mathcal{S}_1}^{L} c_{m'}(t,\omega) - \tilde{c}^{\pi}_m(t,\omega)\big]$ can be bounded by the sum of a $O(\log T)$ term and $\mathbb{P}[\sum_{m\in\tilde{S}_m} Q_m(t) > |\tilde{S}_m|K]$. First, by analyzing the connected components of the slack servers with higher utility than s_m, we can show that the probability of the job dispatchers in those components send jobs to s_m is of order $O(1/T)$, which can be ignored without affecting the regret analysis. Conditioning on those job dispatchers do not send jobs to s_m, we consider the potential function $Z_m(t) = \sum_{m\in\mathcal{S}_m} Q_m(t)$, and same as in the previous proof we can establish a one-slot negative drift argument for $Z_m(t)$ and thus bound $\mathbb{P}[\sum_{m\in\tilde{S}_m} Q_m(t) > |\tilde{S}_m|K]$. Therefore, we have that the Generalized Priority-K policy achieves $O(\log T)$-regret. □

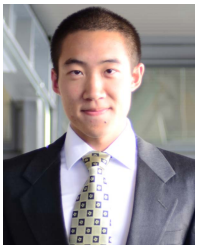
REFERENCES

[1] M. Bramson, Y. Lu, and B. Prabhakar, "Randomized load balancing with general service time distributions," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 38, no. 1, pp. 275–286, 2010.

[2] T. Bonald, M. Jonckheere, and A. Proutiére, "Insensitive load balancing," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 367–377, Jun. 2004.

[3] S. Stidham, "Optimal control of admission to a queueing system," *IEEE Trans. Autom. Control*, vol. AC-30, no. 8, pp. 705–713, Aug. 1985.

[4] V. Gupta, M. H. Balter, K. Sigman, and W. Whitt, "Analysis of join-the-shortest-queue routing for web server farms," *Perform. Eval.*, vol. 64, nos. 9–12, pp. 1062–1081, Oct. 2007.
[5] S. Foss and A. L. Stolyar, "Large-scale join-idle-queue system with general service times," *J. Appl. Probab.*, vol. 54, no. 4, pp. 995–1007, Dec. 2017.
[6] Z. Rosberg and A. M. Makowski, "Optimal routing to parallel heterogeneous servers-small arrival rates," *IEEE Trans. Autom. Control*, vol. 35, no. 7, pp. 789–796, Jul. 1990.
[7] P. Eschenfeldt and D. Gamarnik, "Join the shortest queue with many servers. The heavy-traffic asymptotics," *Math. Oper. Res.*, vol. 43, no. 3, pp. 867–886, Aug. 2018.
[8] M. Mitzenmacher, "The power of two choices in randomized load balancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, pp. 1094–1104, 2001.
[9] Z. Guo et al., "AggreFlow: Achieving power efficiency, load balancing, and quality of service in data center networks," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 17–33, 2020.
[10] J. Fu, B. Moran, J. Guo, E. W. M. Wong, and M. Zukerman, "Asymptotically optimal job assignment for energy-efficient processor-sharing server farms," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 4008–4023, Dec. 2016.
[11] X. Fu and E. Modiano, "Learning-NUM: Network utility maximization with unknown utility functions and queueing delay," *Proc. ACM Mobihoc*, pp. 21–30, Jul. 2021.
[12] X. Fu and E. Modiano, "Elastic job scheduling with unknown utility functions," *Perform. Eval.*, vol. 152, Dec. 2021, Art. no. 102229.
[13] T. Chen and G. B. Giannakis, "Bandit convex optimization for scalable and dynamic IoT management," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 1276–1286, Feb. 2019.
[14] A. Agarwal, D. P. Foster, D. Hsu, S. M. Kakade, and A. Rakhlin, "Stochastic convex optimization with bandit feedback," *SIAM J. Optim.*, vol. 23, no. 1, pp. 213–240, Jan. 2013.
[15] A. L. Stolyar, "Optimal routing in output-queued flexible server systems," *Probab. Eng. Informational Sci.*, vol. 19, no. 2, pp. 141–189, Apr. 2005.
[16] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal power allocation in server farms," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 1, pp. 157–168, Jun. 2009.
[17] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal, "Size-based scheduling to improve web performance," *ACM Trans. Comput. Syst.*, vol. 21, no. 2, pp. 207–233, May 2003.
[18] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. C-35, no. 12, pp. 1347–1356, Dec. 1987.
[19] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, Mar. 1985.
[20] S. Bubeck, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Found. Trends Mach. Learn.*, vol. 5, no. 1, pp. 1–122, 2012.
[21] A. Garivier and O. Cappé, "The KL-UCB algorithm for bounded stochastic bandits and beyond," in *Proc. Conf. Learn. Theory*, 2011, pp. 359–376.
[22] J.-Y. Audibert, R. Munos, and C. Szepesvári, "Exploration–exploitation tradeoff using variance estimates in multi-armed bandits," *Theor. Comput. Sci.*, vol. 410, no. 19, pp. 1876–1902, Apr. 2009.
[23] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 151–159.
[24] Y. Gai, B. Krishnamachari, and R. Jain, "Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation," in *Proc. IEEE Symp. New Frontiers Dyn. Spectr. (DySPAN)*, Apr. 2010, pp. 1–9.
[25] E. Fouché, J. Komiyama, and K. Böhm, "Scaling multi-armed bandit algorithms," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1449–1459.
[26] J. Komiyama, J. Honda, and H. Nakagawa, "Optimal regret analysis of Thompson sampling in stochastic multi-armed bandit problem with multiple plays," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1152–1161.
[27] A. Badanidiyuru, R. Kleinberg, and A. Slivkins, "Bandit with knapsacks," *J. ACM*, vol. 65, no. 3, pp. 1–55, 2018.
[28] C. Jiang and R. Srikant, "Bandits with budgets," in *Proc. 52nd IEEE Conf. Decis. Control*, Dec. 2013, pp. 5345–5350.
[29] S. Cayci, A. Eryilmaz, and R. Srikant, "Budget-constrained bandit over general cost and reward distributions," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 4388–4398.
[30] S. Krishnasamy, R. Sen, R. Johari, and S. Shakkottai, "Learning unknown service rates in queues: A multiarmed bandit approach," *Oper. Res.*, vol. 69, no. 1, pp. 315–330, Jan. 2021.
[31] T. Choudhury, G. Joshi, W. Wang, and S. Shakkottai, "Job dispatching policies for queueing systems with unknown service rates," 2021, *arXiv:2106.04707*.
[32] T. Stahlbuhk, B. Shrader, and E. Modiano, "Learning algorithms for minimizing queue length regret," *IEEE Trans. Inf. Theory*, vol. 67, no. 3, pp. 1759–1781, Mar. 2021.
[33] J. He, D. Zhou, and Q. Gu, "Logarithmic regret for reinforcement learning with linear function approximation," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 4171–4180.
[34] L. Zheng and L. Ratliff, "Constrained upper confidence reinforcement learning," in *Proc. 2nd Conf. Learn. Dyn. Control*, 2020, pp. 620–629.
[35] I. Osband and B. Van Roy, "Near-optimal reinforcement learning in factored MDPs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1–9.
[36] B. Hajek, "Hitting-time and occupation-time bounds implied by drift analysis with applications," *Adv. Appl. Probab.*, vol. 14, no. 3, pp. 502–525, Sep. 1982.



Xinzhe Fu received the B.S. degree (Hons.) in computer science from Shanghai Jiao Tong University in 2017 and the Ph.D. degree in communication networks from MIT in 2022. His research focuses on learning and stochastic optimization in queueing networks.

Eytan Modiano (Fellow, IEEE) received the B.S. degree in electrical engineering and computer science from the University of Connecticut at Storrs in 1986 and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, MD, in 1989 and 1992, respectively.

He is currently the Richard C. Maclaurin Professor with the Department of Aeronautics and Astronautics and the Laboratory for Information and Decision Systems (LIDS), MIT. Prior to joining as the Faculty Member at MIT in 1999, he was a Naval Research Laboratory Fellow from 1987 to 1992, a National Research Council Post-Doctoral Fellow from 1992 to 1993, and a member of the Technical Staff at the MIT Lincoln Laboratory from 1993 and 1999. His research interests include modeling, analysis, and design of communication networks and protocols. In 2020, he received the Infocom Achievement Award for contributions to the analysis and design of cross-layer resource allocation algorithms for wireless, optical, and satellite networks. He is the co-recipient of the Infocom 2018 Best Paper Award, the MobiHoc 2018 Best Paper Award, the MobiHoc 2016 Best Paper Award, the Wiopt 2013 Best Paper Award, and the Sigmetrics 2006 Best Paper Award. He was the Editor-in-Chief for IEEE/ACM TRANSACTIONS ON NETWORKING (2017–2020) and served as an Associate Editor for IEEE TRANSACTIONS ON INFORMATION THEORY and IEEE/ACM TRANSACTIONS ON NETWORKING. He was the Technical Program Co-Chair for IEEE Wiopt 2006, IEEE Infocom 2007, ACM MobiHoc 2007, and DRCN 2015; and the General Co-Chair of Wiopt 2021. He has served on the IEEE Fellow Committee in 2014 and 2015. He is an Associate Fellow of the AIAA.