Job Dispatching Policies for Queueing Systems with Unknown Service Rates

Tuhinangshu Choudhury
Carnegie Mellon University
Pittsburgh, PA

Gauri Joshi
Carnegie Mellon University
Pittsburgh, PA

Weina Wang
Carnegie Mellon University
Pittsburgh, PA

Sanjay Shakkottai
University of Texas - Austin
Austin, TX

ABSTRACT

In multi-server queueing systems where there is no central queue holding all incoming jobs, job dispatching policies are used to assign incoming jobs to the queue at one of the servers. Classic job dispatching policies such as join-the-shortest-queue and shortest expected delay assume that the service rates and queue lengths of the servers are known to the dispatcher. In this work, we tackle the problem of job dispatching without the knowledge of service rates and queue lengths, where the dispatcher can only obtain noisy estimates of the service rates by observing job departures. This problem presents a novel exploration-exploitation trade-off between sending jobs to all the servers to estimate their service rates, and exploiting the currently known fastest servers to minimize the expected queueing delay. We propose a bandit-based exploration policy that learns the service rates from observed job departures. Unlike the standard multi-armed bandit problem where only one out of a finite set of actions is optimal, here the optimal policy requires identifying the optimal fraction of incoming jobs to be sent to each server. We present a regret analysis and simulations to demonstrate the effectiveness of the proposed bandit-based exploration policy.

CCS CONCEPTS

• **Mathematics of computing** → **Queueing theory**; • **Networks** → **Network performance analysis**.

ACM Reference Format:
Tuhinangshu Choudhury, Gauri Joshi, Weina Wang, and Sanjay Shakkottai. 2021. Job Dispatching Policies for Queueing Systems with Unknown Service Rates. In *The Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '21), July 26–29, 2021, Shanghai, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3466772.3467047

1 INTRODUCTION

Traditional queueing models [11, 22] such as M/M/1, M/G/k, G/G/k consist of a single central queue holding incoming jobs and one or more servers that are used to serve those jobs. However, in many applications such as supermarket or airport queues, it is more practical for each server to maintain a separate queue consisting of jobs that are assigned to it. This paradigm calls for the design of job dispatching policies such as join-the-shortest queue (JSQ), shortest expected delay (SED) and least-work-left (LWL) that seek to emulate the delay performance of systems with a single central queue by making the most efficient assignments of jobs to server queues. For example, the JSQ dispatching policy polls the queue lengths at the servers and assigns each job to the shortest queue. In large-scale systems such as computing clusters with tens of thousands of servers, an important consideration is that it can be practically infeasible to poll and maintain status information of all the queues. Therefore, alternatives to join-the-shortest-queue such as the power-of-d-choices (Pod) policy [19, 20, 25] obtain queue length information of only a randomly chosen subset of servers in order to reduce the communication and memory cost.

A common assumption in all the the policies described above is that the service rates at which jobs assigned to each server are served are known to the job dispatcher, or are to be homogeneous across servers, which precludes the need for the dispatcher to know them. In emerging applications such as cloud computing and crowd-sourcing, the servers may not be dedicated to jobs assigned by the dispatcher and may encounter interruptions and service slowdown due to background workload. Therefore, the service rate experienced by the assigned jobs can be unknown, highly variable across servers, and also changing over time. Traditional service-rate-agnostic policies are not effective in such systems and service-rate-aware policies such as join-the-fastest-shortest-queue (JFSQ) cannot be used due to the service rates being unknown.

1.1 Main Contributions and Organization

In this paper, we propose a job dispatching policy that learns the unknown service rates of the servers, while simultaneously seeking to minimize the queueing delay experienced by jobs. This problem is at the intersection of queueing systems and online learning. It sheds light on a novel exploration-exploitation trade-off where the job dispatching policy needs to strike a balance between assigning jobs to all servers in order to better estimate their service rates (exploration) and preferentially sending jobs to the faster servers to minimize the queueing delay experienced by jobs (exploitation).

Unlike classic multi-armed bandits (MABs) where only one of the actions is optimal, in the queueing setting considered in this paper, an optimal policy would typically use several fast servers. Therefore, it is necessary to perform exploration in order to identify the subset of servers that should continue receiving jobs asymptotically. However, more importantly, only identifying this optimal subset of servers is *not enough* for learning an optimal job dispatching policy. We need to also accurately estimate the service rates of servers in this subset. Interestingly, we are able to achieve this by virtue of some special properties of queueing systems. In particular,

MobiHoc '21, July 26–29, 2021, Shanghai, China

ACM ISBN 978-1-4503-8558-9/21/07.
https://doi.org/10.1145/3466772.3467047

after we identify the optimal subset of servers, we exploit by dispatching jobs only to this subset of servers. But meanwhile, since we keep obtaining service time samples from the jobs dispatched to these servers, we also continue to improve the learning accuracy of the service rates. Therefore, exploitation and improvement in estimation are taking place simultaneously in this queueing system.

The rest of the paper is organized as follows. In Section 2 we describe the system model and formulate the problem concretely. In Section 3, we find the optimal weighted random routing policy, which serves as the performance baseline. In Section 4 we propose a bandit-based ϵ_t-exploration algorithm. The distinction between multi-armed bandits and our queueing setting leads to very different regret analysis, presented in Section 5. In Section 6 we demonstrate the effectiveness of the proposed policy via simulations. Due to lack of space, we mainly present proof sketches here, and provide the full proofs in the technical report [6] available online.

1.2 Related Work

Bandits have had a rich history, both from an optimal control perspective (see [18] for a survey) and a finite-time regret perspective (see [16] for a survey). Our paper focuses on bandits in queueing settings – while this intersection has had a rich history (starting from the Klimov's model [12] focusing on optimal control), our focus is on a finite-time regret formulation. At a high-level, the regret perspective formulates queueing problems with unknown statistics (e.g., of the service or arrival processes), with the goal of characterizing the loss/regret in performance of a resource allocation (with learning) algorithm with respect to a genie-policy that has access to the complete statistics. Such regret formulations have recently been introduced both in the adversarial setting [26] and the stochastic setting [14]. Walton [26] has shown that in an adversarial setting, the queue regret (difference between the queue-length induced by a learning algorithm with respect to a static optimal policy) increases at most sublinearly in time. On the other hand, in a stochastic setting, Krishnasamy et al. [14] have shown that the expected regret in fact decreases with time (roughly as $\tilde{O}(1/t)$).

Starting from the above studies, there has been increasing interest in the regret of algorithms in various queueing settings [5, 9, 13, 17, 23]. Krishnasamy et al. [13] studied the problem of scheduling jobs using the $c\mu$ rule with unknown service rates and showed that the cumulative queue regret (i.e., sums of queue regret over time) is $O(1)$. Liu et al. [17] studied the problem of distributing different job classes across servers with unknown rewards for each job class-server pair and proves a reward regret of $O(\sqrt{t})$. Bandits problems of similar flavour are also studied in the communication system settings, where Cayci and Eryilmaz [5] studied channel allocation in wireless downlink systems with unknown channel statistics, with an objective to identify the optimal number of channels to activate and proposed an UCB-based index policy that achieves $O(\ln t)$ regret. The task of selecting the optimal channel in a wireless system with a single transmitter/receiver and multiple channels was studied by Stahlbuhk et al. [23], and they derived queue length based policies that achieve $O(1)$ cumulative queue regret by exploiting samples acquired during the idle time of queues. Finally, Fatale et al. [9] studied regret from an age of information viewpoint.

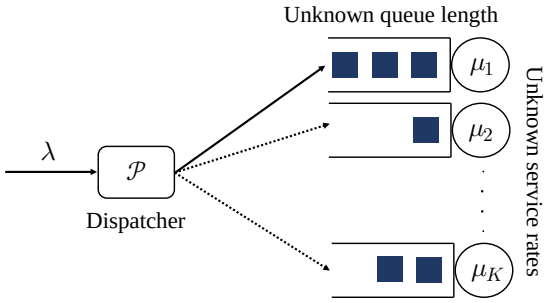

Figure 1: Our system model with job arrival rate λ and K servers with service rates $\mu_1, \ldots, \mu_K$ respectively. The dispatching policy $\mathcal{P}$ assigns each job to one of K queues. The service rates and queue lengths are unknown to the dispatcher; it only observes the service times of departed jobs.

Unlike these studies, our setting is one where the queues are not centrally located at the dispatcher. Instead, jobs are dispatched to individual queues based on partial information; this setting requires both learning a discrete support set and dispatch weights.

2 PROBLEM FORMULATION

2.1 System Model, Arrivals and Departures

We consider a multi-server discrete-time[1] queueing system consisting of K servers, with one queue at each server storing the unfinished jobs that are dispatched to it, as illustrated in Fig. 1. Jobs arrive into the system according to a Bernoulli process with arrival rate λ, where $0 < \lambda < 1$. Specifically, let $A(t)$ denote the number of job arrivals at the beginning of time slot t. Then $A(t) = 1$ with probability λ and $A(t) = 0$ with probability $1 - \lambda$. Incoming jobs are dispatched to one of the the K servers according to a scheduling policy $\mathcal{P}$. Once dispatched, the job joins a first-come-first-served queue with an infinite buffer size at that server. The K servers have geometrically distributed service times with parameter μ_i. That is, after a job reaches the head of the queue at server i, it departs at the end of the next time slot with probability μ_i. With the arrival rate λ and the service of the i-th server being μ_i, the system is stable only if $\lambda < \sum_{i=1}^{K} \mu_i$, a condition that we assume to be true.

Let $A_i(t)$ and $D_i(t)$ denote the number of arrivals to queue i and the number of departures from queue i respectively during time slot t. Let $Q_i(t)$ represent the length of queue i at the beginning of slot t, including the job that is currently in service. We assume that the system starts with empty queues, i.e., $Q_i(0) = 0$, for all i. Then the queue evolution process is given by

$$Q_i(t+1) = Q_i(t) + A_i(t) - D_i(t). \tag{1}$$

We use $X_{i,n}$ to denote the service time of the n-th job that departs from server i. It is the time since the job reaches the head of its queue and starts service until it departs from the system. $X_{i,n}$ is geometrically distributed with success probability μ_i, that is, $\Pr(X_{i,n} = x) = (1-\mu_i)^{x-1}\mu_i$ for $x \in \{1, 2, \ldots\}$.

[1] Although we use the discrete-time assumption for the regret analysis presented in this paper, the proposed policy can be used in continuous time systems. We conjecture that the regret analysis is extendable to continuous time systems as well, but this extension is beyond the scope of this paper and is left for future work.

2.2 Information Available to the Dispatcher

Service Rates and Queue Lengths are Unknown. Job dispatching policies for the multi-server setting described above have been extensively studied in previous literature [11, 22]. However, most prior works assume that the dispatcher knows the service rates $\mu_1, \ldots, \mu_K$, and it also has either full or partial information about the queue lengths $Q_i(t)$. For example, for homogeneous systems where $\mu_1 = \cdots = \mu_K = \mu$, the join-the-shortest-queue (JSQ) policy has full queue information and sends each incoming job to the server $i^* \in \arg\min Q_i(t)$, i.e., the shortest queue, with ties broken at random. Power-of-d-choice (Pod) policies [19, 20, 25] reduce the cost of querying queue lengths by sampling d queues uniformly at random and dispatching the incoming job to the shortest queue. For heterogeneous service rates, JSQ can be generalized to the join-the-shortest-fastest-queue (JSFQ) [8, 27, 28], which breaks ties in favor for the queue with the fastest server. Other policies for systems with heterogeneous servers such as shortest expected delay (SED) [3, 10] also use some form of queue length information to make job dispatch decisions. In contrast, in this work, we consider that the service rates $\mu_1, \mu_2, \ldots, \mu_K$ of the K servers are *heterogeneous and unknown* to the dispatcher. Similarly, the queue lengths $Q_i(t)$ for $i = 1, \ldots, K$ are also *unknown* to the dispatcher.

Dispatcher Observes Service Times of Departed Jobs. In lieu of service rates and queue lengths, we consider that the dispatcher observes service times $X_{i1}, \ldots X_{iN_i(t)}$ of the $N_i(t)$ jobs that depart from server i by time t. In practice, this information can be made available to the dispatcher by having the server send the dispatcher a notification when a job reaches the head of its queue and begins service and another notification when it departs. The dispatching policy $\mathcal{P}$ can use this information to estimate the service rates μ_1, $\ldots \mu_K$. For example, it can estimate the service rate vector $\hat{\boldsymbol{\mu}}(t) = (\hat{\mu}_1(t), \hat{\mu}_2(t), \cdots, \hat{\mu}_K(t))$ at time t, where $\hat{\mu}_i(t)$ is given by

$$\hat{\mu}_i(t) = \frac{N_i(t)}{\sum_{j=1}^{N_i(t)} X_{i,j}}, \tag{2}$$

and use it to dispatch jobs. For instance, it can dispatch a larger fraction of jobs to server with a higher service rate estimate.

2.3 Weighted Random Routing Policies

The service rate estimate $\hat{\mu}_i(t)$ can be used by the dispatcher in a variety of ways to make job dispatch decisions. Among all the possible scheduling policies, we focus on the class of *weighted random routing policies*, which are defined as follows.

Definition 1 (Weighted Random Routing $\mathcal{P}$)**.** In time slot t, the dispatcher associates a probability $\hat{p}_i(t)$ with server i, where $\hat{p}_i(t)$'s satisfy the property $\sum_{j=1}^{K} \hat{p}_j(t) = 1$. We call the probability vector $\hat{\boldsymbol{p}}(t) = (\hat{p}_1(t), \hat{p}_2(t), \cdots, \hat{p}_K(t))$ the *routing vector*. A job that arrives at time t is dispatched to server i with probability $\hat{p}_i(t)$, independent of other jobs. The routing vector $\hat{\boldsymbol{p}}(t) = f(\lambda, \hat{\boldsymbol{\mu}}_i(t))$, where $f : [0,1]^{K+1} \to [0,1]^K$ is a fixed, deterministic function.

The uniform random routing policy corresponds to setting $\hat{\boldsymbol{p}}(t) = (1/K, \cdots, 1/K)$. Since the routing vector $\hat{\boldsymbol{p}}(t)$ is a fixed, predetermined function of $\hat{\mu}_i(t)$, policies such as round-robin dispatching that retain a memory of where past jobs were dispatched are not included in this class of weighted random routing policies.

Optimal Weighted Random Routing. Consider a genie system where the dispatcher knows the service rates $\mu_1, \ldots, \mu_K$. Then the optimal weighted routing policy $\mathcal{P}^*$ is defined as the policy that chooses the optimal $\boldsymbol{p}^* = f(\lambda, \boldsymbol{\mu})$, that minimizes the expected steady-state queue length $\mathbb{E}[\sum_{i=1}^{K} Q_i(\infty)]$, which is equivalent to minimizing the mean response time experienced by incoming jobs. We will derive $\boldsymbol{p}^*$ in Section 3.

2.4 Measuring Performance in terms of Regret

We seek to design a weighted random routing policy $\mathcal{P}$ that starts with no knowledge of the service rates and converges to the optimal random routing policy $\mathcal{P}^*$. To evaluate the transient performance of $\mathcal{P}$ in terms of how quickly it learns $\mathcal{P}^*$, we define a performance metric $\Psi_{\mathcal{P}\mathcal{P}^*}(t)$, referred to as the regret of $\mathcal{P}$. In Section 5, we analyze the performance of our proposed dispatching policy in terms of the expected regret $\mathbb{E}[\Psi_{\mathcal{P}\mathcal{P}^*}(t)]$.

Definition 2 (Regret of a Dispatching Policy)**.** The regret $\Psi_{\mathcal{P}\mathcal{P}^*}$ of a dispatching policy $\mathcal{P}$ with respect to the optimal baseline $\mathcal{P}^*$ is

$$\Psi_{\mathcal{P}\mathcal{P}^*}(t) \triangleq \sum_{\tau=1}^{t} \sum_{i=1}^{K} \left(Q_i(\tau) - Q_i^*(\tau) \right), \tag{3}$$

where $Q_i^*(t)$ represents the queue length at server i at time t when following policy $\mathcal{P}^*$.

The $\Psi_{\mathcal{P}\mathcal{P}^*}(t)$ represents the cost of using policy $\mathcal{P}$ instead of $\mathcal{P}^*$ in terms of cumulative queue length till time t. Note that the cumulative queue length is the total time spent by all jobs that arrived before time t, including the jobs that have departed. Hence, regret represents the additional time jobs stayed in the system when using policy $\mathcal{P}$ instead of $\mathcal{P}^*$. It is the penalty the policy $\mathcal{P}$ has to pay for the lack of knowledge the service rates system.

Difference from the Regret used in Multi-armed Bandits. Although similar, the regret considered in this paper and its analysis is fundamentally different from the multi-armed bandit setting [4, 15, 16]. In the multi-armed bandit setting with K arms, asymptotically optimal algorithms pull the best arm (with the highest mean reward) $O(t)$ times and pull all the $K-1$ sub-optimal arms $O(\log t)$ times. In our queueing setting, the optimal random routing policy generally sends $O(t)$ jobs to all servers with $p_i^* > 0$ and not just the fastest server with the highest service rate μ_i. We seek fast convergence of the routing vector $\hat{\boldsymbol{p}}(t)$ to the optimal $\boldsymbol{p}^*$ so as to dispatch the optimal fraction p_i^* of incoming jobs to each server i.

2.5 Justification for Focusing on Weighted Random Routing Policies

In this section, we justify why we choose to focus on the class of weighted random routing policies. First, we explain why service rates and queue lengths are unknown in the large-scale systems envisioned in this work. Furthermore, we show that even if partial or delayed queue length information is available, using it and performing join-the-shortest-queue (JSQ) or join-the-fast-shortest-queue (JFSQ) dispatching does not give a large performance improvement over weighted random routing.

Why the Service Rates are Unknown. Traditionally, multi-server queueing systems consisted of dedicated servers and a single source

of incoming jobs. However, in modern applications such as cloud data centers, a server may be receiving jobs from several different applications. For instance, it may be running background workload such as check-pointing and garbage collection, or higher priority jobs coming from other sources [7]. As a result, the effective service rate μ_i of server i as seen by the dispatcher of any one application depends on the external workload. Due to privacy constraints and communication delays, it is infeasible for each dispatcher to know and keep track of the external workload at each server. Therefore, we consider that service rates μ_i are unknown to the dispatcher.[2]

Why Queue Lengths are Unknown. In large-scale systems with multiple job sources, each server's queue receives jobs from many dispatchers. In this setting, it is difficult to obtain queue length information due to two reasons: 1) privacy concerns – if a server reveals its total queue length $Q_i(t)$ to one of the dispatcher, it may compromise the privacy of other dispatchers by revealing information about how many jobs they sent to that queue and 2) even if privacy is not a concern, due to large communication delays incurred when a dispatcher queries the queue length of a server, the queue length information may become stale by the time the job is dispatched. Therefore, we consider that the queue lengths $Q_i(t)$ are unknown to the dispatcher.

Limited Utility of Partial or Delayed Queue Length Information. In Fig. 2, we consider a system of 6 servers with service rates μ_i such that $\mu_i = 2^{i-1}\mu_1$, and $\sum_{i=1}^{6} \mu_i = 0.99$. We show a comparison of the mean response times (waiting time in queue plus service time) for optimal weighted random (OWR) routing, which knows the service rates μ_i but does not use queue length information, with JSQ and JFSQ, which use queue length information to make job dispatching decisions. Our goal is to demonstrate that when the queue length information is partial or delayed, a queue-length-agnostic policy such as OWR performs nearly as well as JSQ and JFSQ.

To model partial and delayed queue length information, we consider that apart from the rate λ job arrivals at the dispatcher, server i has external Poisson arrivals at rate λ_i^{ext} that are not visible to the dispatcher. For Fig. 2a1 and Fig. 2b1, we choose $\lambda_i^{ext} = \mu_i/2$, while for Fig. 2a2 and Fig. 2b2, we choose $\lambda_i^{ext} = 4\mu_i/5$. In Fig. 2a, we consider that due to privacy concerns, the dispatcher only has queue length information about the jobs that it sent to each queue, but not about the external arrivals. In Fig. 2b, we consider the case of delayed queue lengths, where the dispatcher receives updated queue length information (including both its jobs and the external arrivals) with probability 1/3. For OWR, we assumed that the policy does not know μ_i or λ_i^{ext} but knows the difference $(\mu_i - \lambda_i^{ext})$ and uses the routing vector $f(\lambda, \boldsymbol{\mu} - \boldsymbol{\lambda}^{ext})$ to dispatch jobs. All of the simulations are averaged over 40 trials of 10^7 job departures each. For both these cases, observe in Fig. 2 that in the low to moderate load regimes (load = $(\lambda + \sum_{i=1}^{K} \lambda_i^{ext})/\sum_{i=1}^{K} \mu_i)$), the optimal weighted random routing (OWR) policy is comparable to JFSQ and better than JSQ in terms of the mean response time. Moreover, observe that as the external arrival rate λ_i^{ext} increased from $\mu_i/2$ to $4\mu_i/5$, the load beyond which OWR performs worse than JFSQ shifted from 0.8 to 0.9 roughly. This is because at heavy load, the cost of a sub-optimal allocation of jobs using a partial or delayed information is more than using no information at all. We conjecture that OWR becomes more useful as the cross traffic load increases.

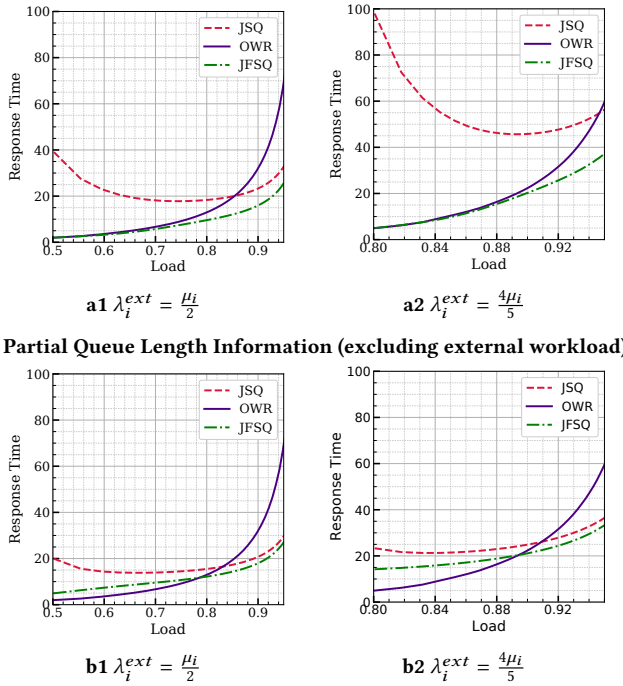

Figure 2: OWR performs better than JSQ and similar to JFSQ in the middle and low traffic regime. In the heavy traffic regime, both JSQ and JFSQ performs better than OWR. OWR performs better as the external workload increases.

As a result of these observations, we choose to focus on the class of weighted random routing policies that do not take into account queue lengths when making job dispatching decisions, and seek to design a dispatching policy that can learn unknown service rates while simultaneously minimizing the regret (see Definition 2). Another reason is that for this class of policies, the optimal policy $\mathcal{P}^*$ that minimizes the steady-state cumulative queue length is clearly defined, as we show in Section 3 below. In contrast, the optimal policy is not known in the case where queue lengths are considered for job assignment decisions. Although policies such as JSQ and JFSQ perform well in practice and in the heavy-traffic regime, it is unclear which policy is optimal in other regimes.

3 OPTIMAL WEIGHTED RANDOM ROUTING POLICY FOR KNOWN SERVICE RATES

We define *optimal weighted random routing (OWR) policy* as the weighted random routing policy that minimizes the mean response time of jobs in steady state or equivalently, the policy that minimizes the cumulative steady-state queue length $\mathbb{E}[\sum_{i=1}^{K} Q_i(\infty)]$. Let $\boldsymbol{p}^* = (p_1^*, p_2^*, \cdots, p_K^*)$ be the routing vector corresponding to the optimal weighted routing policy as a function of arrival rate and the service rates and we will refer to it as the *optimal routing vector*.

[2]In practice, the effective service rates may vary over time depending on the external workload. For tractability of the analysis, we do not consider time-varying service rates μ_i. However, the estimates in (2) can be modified to discount older service time observations in order to account for time-varying service rates.

For a $Geo/Geo/1$ queueing system with arrival rate λ' and departure rate μ', the expected steady-state queue length, $\mathbb{E}[Q(\infty)]$, is given by (see [22] Chapter 3 for the derivation)

$$\mathbb{E}[Q(\infty)] = \frac{\lambda'(1-\mu')}{\mu'-\lambda'}. \tag{4}$$

Now, for our system with arrival rate λ and service rates μ_i's, if every incoming job is assigned to server i with probability p_i, the system can be viewed as K $Geo/Geo/1$ queues each with arrival rate $p_i\lambda$ and service rate μ_i respectively. Hence, the steady state queue length of the system is given by

$$\mathbb{E}\left[\sum_{i=1}^{K} Q_i(\infty)\right] = \sum_{i=1}^{K} \mathbb{E}[Q_i(\infty)] = \sum_{i=1}^{K} \frac{p_i\lambda(1-\mu_i)}{\mu_i - \lambda p_i}. \tag{5}$$

Hence, the optimal routing vector would be the solution to the following optimization problem:

$$\min_{p_1,p_2,\cdots,p_K} \quad \sum_{i=1}^{K} \frac{p_i\lambda(1-\mu_i)}{\mu_i-\lambda p_i} \tag{6}$$

$$\text{s.t.} \quad \sum_{i=1}^{K} p_i = 1 \tag{7}$$

$$p_i \geq 0, \quad p_i\lambda < \mu_i, \ \forall i, \tag{8}$$

where the constraint $p_i\lambda < \mu_i$ ensures stability of the queue i and the remaining constraints ensure that $\mathbf{p}$ forms a valid routing vector. We show in [6] that the optimal routing vector is given by $\boldsymbol{p}^* = f(\lambda, \boldsymbol{\mu})$, where $f : [0,1]^{K+1} \to [0,1]^K$ is a function given by

$$f(\lambda,\boldsymbol{\mu})(i) = \begin{cases} \frac{\mu_i}{\lambda} - \frac{\sqrt{\mu_i(1-\mu_i)}}{\sum_{j\in\mathcal{S}(\lambda,\boldsymbol{\mu})}\sqrt{\mu_j(1-\mu_j)}}\left(\frac{\sum_{j\in\mathcal{S}(\lambda,\boldsymbol{\mu})}\mu_j}{\lambda}-1\right), & \text{if } i\in\mathcal{S}(\lambda,\boldsymbol{\mu}), \\ 0, & \text{if } i\notin\mathcal{S}(\lambda,\boldsymbol{\mu}). \end{cases} \tag{9}$$

where $\mathcal{S}(\lambda,\boldsymbol{\mu})$ is a subset of servers which we refer to as the *optimal support set* such that $i \in \mathcal{S}(\lambda,\boldsymbol{\mu})$ if and only if $p_i^* > 0$.

In [6], we formally prove that if $\mu_i \geq \mu_j$, then $p_i^* \geq p_j^*$. Therefore, only the slowest servers are excluded from the support set $\mathcal{S}(\lambda,\boldsymbol{\mu})$, and the support set has to be $[1,\ldots,i]$ for some $i \in \{1,2,\ldots K\}$. To find the optimal set $\mathcal{S}(\lambda,\boldsymbol{\mu})$ and the optimal routing vector $\boldsymbol{p}^*$ we can use the iterative algorithm given below.

(i) Initialize the support set $\mathcal{S}(\lambda,\boldsymbol{\mu}) = \{1,2,\ldots,K\}$.
(ii) Calculate p_i^* according to (9).
(iii) If $p_i^* < 0$ for any $i \in \{1,\ldots,K\}$ or $p_i^* = 0$ for any $i \in \mathcal{S}(\lambda,\boldsymbol{\mu})$, then remove the slowest server $\arg\min_{i\in\mathcal{S}(\lambda,\boldsymbol{\mu})}\mu_i$ from the support set and repeat from Steps (ii) and (iii).

The proof of the correctness of this algorithm is given in [6].

4 PROPOSED JOB DISPATCHING POLICY FOR UNKNOWN SERVICE RATES

Recall that we consider a dispatcher who does not know the service rate vector $\boldsymbol{\mu} = (\mu_1,\ldots,\mu_K)$ and thus relies on the estimated service rate vector $\hat{\boldsymbol{\mu}}(t) = (\hat{\mu}_1(t),\ldots,\hat{\mu}_K(t))$. Our goal is to design a dispatching policy $\mathcal{P}$ that minimizes the expected regret $\mathbb{E}[\Psi_{\mathcal{P}\mathcal{P}^*}(t)]$ with respect to the optimal weighted random routing policy $\mathcal{P}^*$.

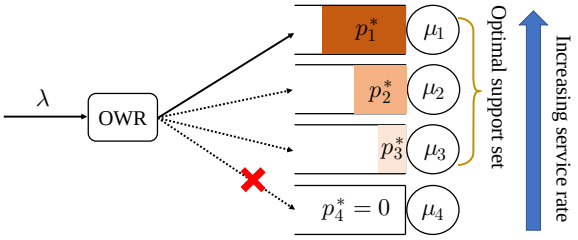

Figure 3: An illustration of the optimal weighted random routing policy. The job arrival rate is λ is split and there are 4 servers with service rates $\mu_1,\ldots,\mu_4$ respectively such that $\mu_1 \geq \mu_2 \geq \mu_3 \geq \mu_4$.

In order to asymptotically converge to $\mathcal{P}^*$, it is important for the dispatching policy to correctly identify the optimal support set $\mathcal{S}(\lambda,\boldsymbol{\mu})$ for which $p_i^* > 0$. If a server with $p_i^* > 0$ is excluded from the estimated support set, then the dispatcher will not send any jobs to it and hence cannot converge to the optimal policy $\mathcal{P}^*$. In this section we first demonstrate that it is necessary to explore (that is, send jobs to all K servers) infinitely often for achieving a reasonable regret in Section 4.1. We then present our policy in Section 4.2.

4.1 The Necessity of Exploration

It is well-known that for stochastic multi-armed bandits problems, it is necessary to explore infinitely often to achieve an optimal regret. Interestingly, there are recent results showing that no explicit exploration is needed for achieving an optimal regret in some queueing systems with unknown parameters [13]. There the exploration comes for free when running a stabilizing policy. However, for the job dispatching problem we consider in this paper, we demonstrate below that infinitely often exploration is still necessary.

A naive dispatching policy may dedicate a constant amount of time at the beginning to exploration to obtain a good estimate of the service rate vector. Then after the initial exploration phase, the policy uses the estimate $\hat{\boldsymbol{\mu}}(t)$ at every time slot t to compute the routing vector and dispatches arriving jobs accordingly, while keeping updating the estimate $\hat{\boldsymbol{\mu}}(t)$ using the service times of completed jobs. We will construct an example to demonstrate that this can lead to a situation where a server in the optimal support set $\mathcal{S}(\lambda,\boldsymbol{\mu})$ is forever excluded from the estimated optimal set, which will incur a linear regret.

Consider a two-server system with $\lambda = 0.2$ and $\boldsymbol{\mu} = (0.45, 0.55)$. One can verify that the optimal routing vector is $(0.25, 0.75)$. Suppose for the first n time slots, the dispatching policy assigns an arriving job to one of the two servers chosen uniformly at random. We consider an event $\mathcal{E}$ defined by the scenario below.

Suppose that there are k job arrivals during the first n time slots to server 1. Let t' be the earliest time by which all the k jobs that are assigned to server 1 have departed. Then we consider the scenario where the estimated service rates satisfy that $\hat{\mu}_1(t) \leq 0.05$ for all time t with $t \leq t'$ and $\hat{\mu}_2(t) \geq 0.25$ for all time t.

We first argue that under the event $\mathcal{E}$, the regret scales linearly with time. Note that under $\mathcal{E}$, for any time t with $t \leq t'$, the routing vector is $(0,1)$ since $\hat{\mu}_1(t) \leq 0.05$ and $\hat{\mu}_2(t) \geq 0.25$. At time $t = t'+1$,

Algorithm 1 An ϵ_t-Exploration Policy for Learning Optimal Weighted Random Routing

while $t \geq 0$ **do**
  if a job arrives **then**
    $\chi(t) \leftarrow$a Bernoulli sample with mean $\min\left\{1, \frac{K \ln t}{t}\right\}$
    if $\chi(t) = 1$ **then** ▷ Explore
      Dispatch to one of the servers uniformly at random
    else ▷ Exploit
      Compute routing vector $\hat{\boldsymbol{p}}(t) = f(\lambda, \hat{\boldsymbol{\mu}}(t))$
      Dispatch to server i with probability $\hat{p}_i(t)$
    end if
  end if
  for $i = 1, 2, \cdots K$ **do**
    if a job departs from server i **then**
      Update $\hat{\mu}_i(t)$ using (2)
    end if
  end for
end while

we still have that $\hat{\mu}_1(t) \leq 0.05$ since no new job is sent to server 1 and that $\hat{\mu}_2(t) \geq 0.25$ by the definition of event $\mathcal{E}$, which makes the routing vector remain $(0, 1)$. Repeating this argument for all the time slots after $t' + 1$ we can see that for rest of the time no job will be sent to server 1 at all. Now the expected steady-state queue length when using only server 2 is $9/35$, while the expected steady-state queue length using the optimal routing vector is $19/80$. Thus the regret scales roughly as $11t/560$, which is linear in t.

We next show that the event $\mathcal{E}$ happens with a strictly positive probability:

$$\begin{aligned}
\mathbb{P}(\mathcal{E}) &= \left(\sum_{k=1}^{n} \binom{n}{k} (0.5\lambda)^k (1-0.5\lambda)^{n-k} \mathbb{P}\left(\begin{smallmatrix}\text{all } k \text{ jobs to server 1}\\ \text{have service time} \geq 20\end{smallmatrix}\right)\right) \\
&\quad \cdot \mathbb{P}\left(\cap_{t=1}^{\infty} \{\hat{\mu}_2(t) \geq 0.25\}\right) \qquad (10) \\
&\geq \left(1 - \sum_{t=1}^{\infty} \mathbb{P}\left(\hat{\mu}_2(t) < 0.25\right)\right) \\
&\quad \cdot \sum_{k=1}^{n} \binom{n}{k} (0.5\lambda)^k (1-0.5\lambda)^{n-k} \left((0.45)^{20}\right)^k \qquad (11) \\
&\geq 0.09 \sum_{k=1}^{n} \binom{n}{k} (0.5\lambda)^k (1-0.5\lambda)^{n-k} \left((0.45)^{20}\right)^k, \qquad (12)
\end{aligned}$$

which is strictly positive, where (12) is due to Chernoff bound.

4.2 An ϵ_t-Exploration Policy

As demonstrated in Section 4.1, exploring for a fixed amount of time can lead to a linear regret. To ensure enough exploration, we propose an ϵ_t-exploration policy, which explores with probability $\epsilon_t = \frac{K \ln t}{t}$ at each time slot t. When not exploring, the policy treats the estimated service rates as if they were the actual service rates and calculates the optimal routing probabilities based on them; i.e., when not exploring, the policy uses $\hat{\boldsymbol{p}}(t) = (\hat{p}_1(t), \hat{p}_2(t), \cdots, \hat{p}_K(t))$ to make a routing decision at time t. The pseudo-code is presented in Algorithm 1.

5 REGRET ANALYSIS

In this section, we prove an upper bound on the expected regret $\mathbb{E}\left[\Psi_{\mathcal{P}\mathcal{P}^*}(t)\right]$ of the ϵ_t-exploration policy.

To state our upper bound, we first define a quantity $\Delta_{\mathcal{S}}$ that we refer to as the *tolerance gap*, which is analogous to the suboptimality gap for multi-armed bandits. Specifically, let

$$\begin{aligned}
\Delta_{\mathcal{S}} := \sup \big\{ \delta \geq 0 \colon \mathcal{S}\left(\lambda, \boldsymbol{\mu}'\right) = \mathcal{S}(\lambda, \boldsymbol{\mu}) \\
\forall \boldsymbol{\mu}' \text{s.t. } \left|\mu_i' - \mu_i\right| \leq \delta, \forall i \big\},
\end{aligned} \qquad (13)$$

where recall that $\mathcal{S}(\lambda, \boldsymbol{\mu})$ is the optimal support set computed from arrival rate λ and service rate vector $\boldsymbol{\mu}$.

The tolerance gap $\Delta_{\mathcal{S}}$ quantifies how much error in the service rates can be tolerated without incurring a discrepancy in the support set. We can think of the $\boldsymbol{\mu}'$ in (13) as the estimated service rate vector. If $\Delta_{\mathcal{S}} = 0$, then even a slight imprecision in the estimated service rates would make the estimated support set deviate from the optimal support set, indicating hardness of the problem. Therefore, we make the assumption that $\Delta_{\mathcal{S}} > 0$ in our upper bound. We comment that from a practical perspective, this is a very mild assumption since only a small set (with zero measure) in the parameter space $\left\{(\lambda, \boldsymbol{\mu}) \in \mathbb{R}_+^{K+1} \colon \lambda < \sum_{i=1}^{K} \mu_i\right\}$ will violate this assumption.

Theorem 1 (Upper Bound on Expected Regret)**.** Consider a system with arrival rate λ and service rate vector $\boldsymbol{\mu}$ and assume that $\Delta_{\mathcal{S}} > 0$. Then there exists a constant k_1 and a t_0 such that for all $t \geq t_0$, the ϵ_t-exploration policy (Algorithm 1) has an expected regret $\mathbb{E}\left[\Psi_{\mathcal{P}\mathcal{P}^*}(t)\right]$ that admits the following upper bound:

$$\begin{aligned}
&\mathbb{E}\left[\Psi_{\mathcal{P}\mathcal{P}^*}(t)\right] \\
&\quad \leq \sum_{\tau=t_0}^{t-1} \left(\sum_{i:p_i^*>0} \frac{66 k_1 \ln \tau}{r_i^2} \sqrt{\frac{\ln \tau}{\tau}} + \sum_{i=1}^{K} \frac{132 \ln^2 \tau}{r_i^2 \tau} \right) + O(1), \qquad (14)
\end{aligned}$$

where r_i is the residual capacity of server i under the optimal weighted random routing given by $r_i = \mu_i - \lambda p_i^*$.

In the regret upper bound in Theorem 1 above, the first summand $\sum_{\tau=t_0}^{t-1} \sum_{i:p_i^*>0} \frac{66 k_1 \ln \tau}{r_i^2} \sqrt{\frac{\ln \tau}{\tau}}$ is the dominant term and it comes from the estimation error in the estimated routing probability vector $\hat{\boldsymbol{p}}(t)$, and the second summand $\sum_{\tau=t_0}^{t-1} \sum_{i=1}^{K} \frac{132 \ln^2 \tau}{r_i^2 \tau}$ results from the exploration used by the ϵ_t-exploration policy. We note that this regret bound becomes smaller in low to moderate traffic regimes where the residual capacities r_i's are large and the size of the optimal support set $\mathcal{S}(\lambda, \boldsymbol{\mu}) = \{i \colon p_i^* > 0, i = 1, 2, \ldots, K\}$ is small.

In the following subsections, we first couple our system with the system that runs the the optimal weighted random routing policy in Section 5.1 to facilitate the regret analysis. We then prove Theorem 1 in Section 5.2, using several lemmas whose proof sketches are given in Section 5.3.

5.1 Coupling with the Optimal Weighted Random Routing

Consider the system that runs the optimal weighted random routing policy $\mathcal{P}^*$, which we refer to as the *optimal system*. We will annotate quantities in the optimal system with the superscript *, e.g., $A^*(t)$ denotes the total number of job arrivals at time t to the

optimal system, and $Q_i^*(t)$ denotes the length of queue i under $\mathcal{P}^*$. Correspondingly, recall that the regret at time t is defined as

$$\Psi_{\mathcal{P}\mathcal{P}^*}(t) = \sum_{\tau=1}^{t-1}\sum_{i=1}^{K}\left(Q_i(\tau) - Q_i^*(\tau)\right).$$

We assume that the optimal system also starts from empty queues, i.e., $Q_i^*(0) = 0$ for all i.

We couple the system that runs our proposed ϵ_t-exploration policy with the optimal system in the following way.

Arrivals. We couple arrivals such that $A(t) = A^*(t)$ for all time t.

Service. For each server i, let $S_i(t)$ for $t = 0, 1, 2, \ldots$ be i.i.d. Bernoulli random variables that take the value 1 with probability μ_i. We will refer to $S_i(t)$ as the *service* offered by server i at t. If $Q_i(t)+A_i(t) > 0$, we let $D_i(t) = S_i(t)$, where recall that $D_i(t)$ is the number of departures from queue i at t; otherwise it is clear that $D_i(t) = 0$. Similarly, let $S_i^*(t)$ denote the corresponding service offered in the optimal system. We couple the service processes such that $S_i(t) = S_i^*(t)$ for all server i and all time t.

Assignment process. Recall that in the ϵ_t-exploration policy, for each time slot t, with probability $\frac{K\ln t}{t}$ we explore and otherwise we dispatch the arriving job according to the routing vector $\hat{\boldsymbol{p}}(t)$. We couple the dispatching decision generated from $\hat{\boldsymbol{p}}(t)$ with the dispatching decision generated from the optimal routing vector $\boldsymbol{p}^*$ in the optimal system as follows.

For simplicity, we can assume that for each time slot t, we generate a dispatching decision from $\hat{\boldsymbol{p}}(t)$, although this dispatching decision is needed only when there is a job arrival at t and the ϵ_t-exploration policy chooses to exploit. Let the dispatching decision generated from $\hat{\boldsymbol{p}}(t)$ be represented by the server that an arriving job will be dispatched to, denoted as $\sigma(t)$. Then $\sigma(t)$'s probability mass function (pmf) is $\hat{\boldsymbol{p}}(t)$. Similarly, let $\sigma^*(t)$ be the dispatching decision in the optimal system, and then $\sigma^*(t)$'s pmf is $\boldsymbol{p}^*$. Then we couple $\sigma(t)$ and $\sigma^*(t)$ such that they have the following joint pmf:

$$\mathbb{P}(\sigma(t) = i, \sigma^*(t) = j) = \begin{cases} \min\{\hat{p}_i(t), p_i^*\} & \text{if } i = j, \\ \frac{(\hat{p}_i(t) - \min\{\hat{p}_i(t), p_i^*\})\left(p_j^*(t) - \min\left\{\hat{p}_j(t), p_j^*\right\}\right)}{d_{TV}(\hat{\boldsymbol{p}}(t), \boldsymbol{p}^*)} & \text{if } i \neq j, \end{cases} \tag{15}$$

where $d_{TV}(\hat{\boldsymbol{p}}(t), \boldsymbol{p}^*)$ is the total variation distance between $\hat{\boldsymbol{p}}(t)$ and $\boldsymbol{p}^*$ and is given by $d_{TV}(\hat{\boldsymbol{p}}(t), \boldsymbol{p}^*) = \sum_{j=1}^{K}\left(p_j^*(t) - \min\left\{\hat{p}_j(t), p_j^*\right\}\right)$. This coupling is known as the *maximal coupling* (see, e.g., [21]) and it guarantees that $\mathbb{P}(\sigma(t) \neq \sigma^*(t)) = d_{TV}(\hat{\boldsymbol{p}}(t), \boldsymbol{p}^*)$.

With this coupling, we can quantify the probability for a mismatched dispatching decision between our system and the optimal system. In our system, recall that $A_i(t)$ denotes the number of jobs dispatched to server i at time t. We now make a finer distinction between jobs dispatched through exploration and through exploitation under the ϵ_t-exploration policy. Let $A_i^{(E)}(t)$ and $A_i^{(O)}(t)$ denote the numbers of jobs dispatched to server i through exploration and exploitation, respectively. Then $A_i(t) = A_i^{(E)}(t) + A_i^{(O)}(t)$. Lemma 1 below upper-bounds the probability for the mismatch that $A_i^{(O)}(t) = 1, A_i^*(t) = 0$ with the distance $\left|\hat{p}_i(t) - p_i^*\right|$, implying that once the estimates $\hat{p}_i(t)$'s are close to p_i^*'s, the probability of such a mismatch is small.

Lemma 1. For any time slot t and any server i,

$$\mathbb{P}\left[A_i^{(O)}(t) = 1, A_i^*(t) = 0 \;\middle|\; \hat{p}_i(t)\right] \leq \left|\hat{p}_i(t) - p_i^*\right|.$$

Proof of the lemma is given in [6].

5.2 Proof of Regret Bound (Theorem 1)

In this section we prove the upper bound in Theorem 1 on the expected regret $\mathbb{E}\left[\Psi_{\mathcal{P}\mathcal{P}^*}(t)\right] = \mathbb{E}\left[\sum_{\tau=1}^{t-1}\sum_{i=1}^{K}\left(Q_i(\tau) - Q_i^*(\tau)\right)\right]$ based on several lemmas. Proof sketches of these lemmas will be given in Section 5.3, and the detailed proofs are presented in [6].

We first note that the difference between $Q_i(t)$ and $Q_i^*(t)$ can be written in the following recursive form for any t and $\tau \leq t$:

$$Q_i(t) - Q_i^*(t) = Q_i(\tau) - Q_i^*(\tau) + \sum_{\ell=\tau}^{t-1}\left(A_i(\ell) - D_i(\ell) - \left(A_i^*(\ell) - D_i^*(\ell)\right)\right).$$

In this proof, we will consider a specific τ that is the last time queue i is empty. In particular, define $B_i(t)$ as the length of the current busy cycle period as seen at time t, i.e.,

$$B_i(t) = \min\{s \geq 0 : Q_i(t-s) = 0\}. \tag{16}$$

Then it is easy to see that for $\tau = t - B_i(t)$, we have $Q_i(\tau) = 0$ and $Q_i^*(\tau) \geq 0$. In addition, for any ℓ with $\tau \leq \ell \leq t-1$, we have $D_i(\ell) = S_i(\ell)$ since $Q_i(\ell) > 0$. Based on this choice of τ, the queue length difference can be bounded as follows:

$$\begin{aligned} Q_i(t) - Q_i^*(t) &= Q_i(t - B_i(t)) - Q_i^*(t - B_i(t)) + \sum_{\ell=t-B_i(t)}^{t-1}\left(A_i(\ell) - D_i(\ell) - \left(A_i^*(\ell) - D_i^*(\ell)\right)\right) \\ &\leq \sum_{\ell=t-B_i(t)}^{t-1}\left(A_i(\ell) - A_i^*(\ell)\right) + \sum_{\ell=t-B_i(t)}^{t-1}\left(D_i^*(\ell) - D_i(\ell)\right) \\ &\leq \sum_{\ell=t-B_i(t)}^{t-1}\left(A_i^{(E)}(\ell) + A_i^{(O)}(\ell) - A_i^*(\ell)\right) + \sum_{\ell=t-B_i(t)}^{t-1}\left(S_i^*(\ell) - S_i(\ell)\right) \qquad (17) \\ &= \sum_{\ell=t-B_i(t)}^{t-1} A_i^{(E)}(\ell) + \sum_{\ell=t-B_i(t)}^{t-1}\left(A_i^{(O)}(\ell) - A_i^*(\ell)\right) \qquad (18) \\ &\leq \sum_{\ell=t-B_i(t)}^{t-1} A_i^{(E)}(\ell) + \sum_{\ell=t-B_i(t)}^{t-1}\mathbf{1}_{\left\{A_i^{(O)}(\ell)=1, A_i^*(\ell)=0\right\}}, \qquad (19) \end{aligned}$$

where (17) uses the facts that $A_i(\ell) = A_i^{(E)}(\ell) + A_i^{(O)}(\ell)$, $D_i^*(\ell) \leq S_i^*(\ell)$, and $D_i(\ell) = S_i(\ell)$; (18) is due to our coupling $S_i(\ell) = S_i^*(\ell)$.

In the upper bound (19) on the queue length difference, the first summand comes from exploration. Since we know that by our ϵ_t-exploration, we have $\mathbb{E}[A_i^{(E)}(\ell)] = \frac{K\ln\ell}{\ell}$, this summand can be properly bounded if we obtain a suitable upper bound on $B_i(t)$. The second summand in (19) comes from exploitation, and it can be

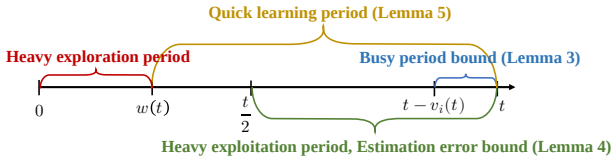

Figure 4: Time structure of lemmas in proof of Theorem 1.

bounded with the estimation error through Lemma 1. To formalize the above intuition, we define the following events:

$$\mathcal{E}_1(t) := \{B_i(t) \le v_i(t), \forall i\}, \text{ where } v_i(t) = \frac{66 \ln t}{r_i^2}, \text{ and} \quad (20)$$

$$\mathcal{E}_2(t) := \left\{\left|\hat{p}_i(\tau) - p_i^*\right| \le k_1 \min\left\{\sqrt{\frac{\ln t}{t}}, p_i^*\right\}, \forall \tau \in \left[\frac{t}{2}+1, t\right], \forall i\right\}, \quad (21)$$

where k_1 is a properly chosen constant. Utilizing these two events, Lemma 2 below establishes an upper bound on the expected queue length difference, which enables us to further bound the regret by analyzing the busy period and the estimation error in service rates.

Lemma 2. There exists a t_0 such that for any time $t \ge t_0$, the total expected queue length difference can be bounded as

$$\sum_{i=1}^{K} \mathbb{E}\left[Q_i(t) - Q_i^*(t)\right] \le \sum_{i=1}^{K} \frac{2v_i(t)\ln t}{t} + \sum_{i:p_i^*>0} k_1 v_i(t)\sqrt{\frac{\ln t}{t}} + t\mathbb{P}\left((\mathcal{E}_1(t))^c\right) + 2t\mathbb{P}\left((\mathcal{E}_2(t))^c\right). \quad (22)$$

With Lemma 2, to bound the expected regret, now it suffices to bound the probabilities $\mathbb{P}\left((\mathcal{E}_1(t))^c\right)$ and $\mathbb{P}\left((\mathcal{E}_2(t))^c\right)$, which are established in Lemmas 3 and 4 below. We demonstrate the time structure of the lemmas in Figure 4.

Lemma 3 (Busy Period Bound)**.** There exist a constant k_2 and a t_0 such that for any $t \ge t_0$, the event $\mathcal{E}_1(t)$ defined in (20) satisfies

$$\mathbb{P}\left((\mathcal{E}_1(t))^c\right) \le 4K\left(\frac{1}{t^7} + \frac{k_2+1}{t^3} + \frac{1}{t^4}\right). \quad (23)$$

Lemma 4 (Estimation Error Bound)**.** There exist a constant k_2 and a t_0 such that for any $t \ge t_0$, the event $\mathcal{E}_2(t)$ defined in (21) satisfies

$$\mathbb{P}\left((\mathcal{E}_2(t))^c\right) \le K\left(\frac{1}{t^7} + \frac{k_2+1}{t^3}\right) + \sum_{i:p_i^*>0}\left(\frac{1}{t^3} + t\exp\left(-\frac{p_i^*\lambda t}{128}\right) + t\exp\left(-\frac{r_i^2 t}{4}\right)\right). \quad (24)$$

Finally, we choose a common t_0 for Lemmas 2–4 and a common k_2 for Lemmas 3 and 4, and put Lemmas 2–4 together to get

$$\begin{aligned}
&\mathbb{E}\left[\Psi_{\mathcal{P}\mathcal{P}^*}(t)\right] \\
&= \sum_{\tau=1}^{t_0-1}\sum_{i=1}^{K}\mathbb{E}\left[Q_i(\tau) - Q_i^*(\tau)\right] + \sum_{\tau=t_0}^{t-1}\sum_{i=1}^{K}\mathbb{E}\left[Q_i(\tau) - Q_i^*(\tau)\right] \\
&\le t_0^2 + \sum_{\tau=t_0}^{t-1}\sum_{i=1}^{K}\frac{2v_i(\tau)\ln\tau}{\tau} + \sum_{\tau=t_0}^{t-1}\sum_{i:p_i^*>0} k_1 v_i(\tau)\sqrt{\frac{\ln\tau}{\tau}} \\
&\quad + \sum_{\tau=t_0}^{t-1}\tau\cdot K\left(\frac{4}{\tau^7} + \frac{4k_2+4}{\tau^3} + \frac{4}{\tau^4}\right) + \sum_{\tau=t_0}^{t-1} 2\tau\cdot K\left(\frac{1}{\tau^7} + \frac{k_2+1}{\tau^3}\right) \\
&\quad + \sum_{\tau=t_0}^{t-1} 2\tau\cdot\sum_{i:p_i^*>0}\left(\frac{1}{\tau^3} + \tau\exp\left(-\frac{p_i^*\lambda\tau}{128}\right) + \tau\exp\left(-\frac{r_i^2\tau}{4}\right)\right) \quad (25) \\
&= t_0^2 + \sum_{\tau=t_0}^{t-1}\sum_{i=1}^{K}\frac{132\ln^2\tau}{r_i^2\tau} + \sum_{\tau=t_0}^{t-1}\sum_{i:p_i^*>0}\frac{66k_1\ln\tau}{r_i^2}\sqrt{\frac{\ln\tau}{\tau}} \\
&\quad + K\sum_{\tau=t_0}^{t-1}\left(\frac{6}{\tau^6} + \frac{6k_2+6}{\tau^2} + \frac{4}{\tau^3}\right) \\
&\quad + 2\sum_{\tau=t_0}^{t-1}\sum_{i:p_i^*>0}\left(\frac{1}{\tau^2} + \tau^2\exp\left(-\frac{p_i^*\lambda\tau}{128}\right) + \tau^2\exp\left(-\frac{r_i^2\tau}{4}\right)\right) \quad (26) \\
&= \sum_{\tau=t_0}^{t-1}\left(\sum_{i:p_i^*>0}\frac{66k_1\ln\tau}{r_i^2}\sqrt{\frac{\ln\tau}{\tau}} + \sum_{i=1}^{K}\frac{132\ln^2\tau}{r_i^2\tau}\right) + O(1),
\end{aligned}$$

which completes the proof of Theorem 1. □

Remark. Our proof techniques used the *absolute* difference in the routing probabilities to analyze the difference in queue lengths. We comment that it might be possible for one to prove a tighter regret upper bound by considering the actual difference in routing probabilities, but the analysis will become much more challenging. Specifically, inaccurate routing probabilities can actually instantaneously benefit the queues whose $\hat{p}_i(t)$ is smaller than the optimal p_i^*. This is a phenomenon not seen in multi-armed bandit problems, and it is worth further investigation.

5.3 Proof Sketches of Lemmas 2–4

The detailed proofs of Lemmas 2, 3, 4 and 5 are presented in [6].

Lemma 2 can be proven using a series of conditioning on event $\mathcal{E}_1(t)$, event $\mathcal{E}_2(t)$ and also the estimated $\hat{\boldsymbol{p}}(t)$. Then the construction of $\mathcal{E}_1(t)$ gives an upper bound on the busy period, Lemma 1 helps translate the number of mismatches into the error in estimation, and finally the construction of $\mathcal{E}_2(t)$ upper bounds the estimation error.

Next for Lemmas 3 and 4, we will just highlight a key lemma used in their proofs, presented as Lemma 5. Lemma 5 below states that for any large enough time t, the estimated optimal support set, $\mathcal{S}(\lambda, \hat{\boldsymbol{\mu}}(t))$, is correct (i.e., equal to the true optimal support set $\mathcal{S}(\lambda, \boldsymbol{\mu})$) for a period of time $[w(t), t]$ with high probability, where $w(t) = 2\exp\left(\Theta\left(\sqrt{\ln t}\right)\right)$. Moreover, during this period of time, the rate at which we dispatch jobs to each server i lies between $\lambda p_i^*/2$ and $\mu_i - r_i/2$; i.e., the arrival rate to each server is no smaller than half of the rate under the optimal weighted random routing, but still leaves at least half of residual capacity under the optimal weighted random routing. We call the period of time $[w(t), t]$ the *quick learning period* since we have "locked" the correct support set and spend all exploitation jobs on learning the service rates of servers in the correct support set. This time structure of Lemma 5 is also illustrated in Figure 4.

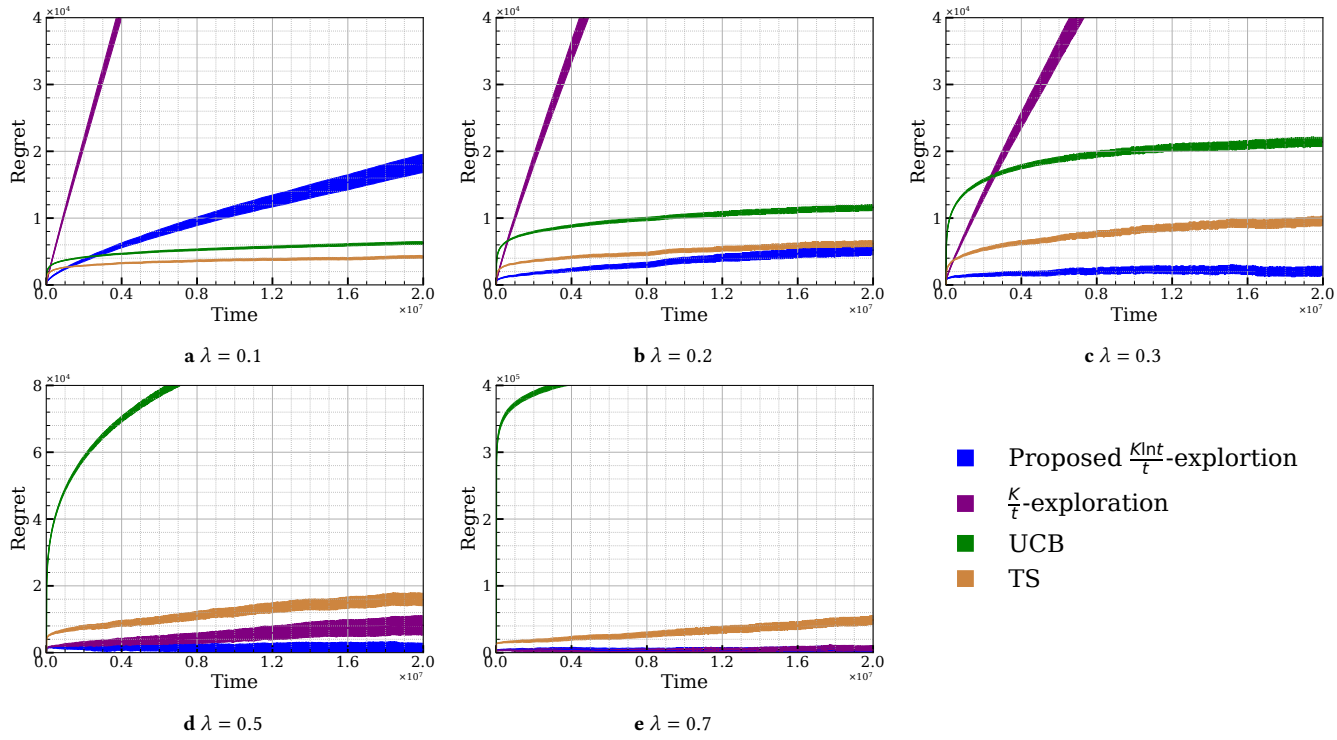

Figure 5: Regret vs time for various policies. The shaded region represent $\pm 2\sigma$ boundary of the mean regret. TS and UCB performs well in load load regime, while K/t-greedy exploration policy performs well in higher traffic regimes. Our policy moderately in very low load regime and performs well in higher traffic regimes.

Lemma 5 (Quick Learning Period). Define the event $\mathcal{E}_3(t)$ as

$$\mathcal{E}_3(t) := \mathcal{E}_{31}(t) \cap \mathcal{E}_{32}(t), \text{ where} \tag{27}$$

$$\mathcal{E}_{31}(t) := \{\mathcal{S}(\lambda, \hat{\boldsymbol{\mu}}(\tau)) = \mathcal{S}(\lambda, \boldsymbol{\mu}), \forall \tau \in [w(t)+1, t]\},$$

$$\mathcal{E}_{32}(t) := \left\{ \frac{\lambda p_i^*}{2} \le \mathbb{E}\left(A_i(\tau) \mid \hat{p}_i(\tau)\right) \le \mu_i - \frac{r_i}{2}, \forall i, \forall \tau \in [w(t)+1, t] \right\}.$$

Then there exist a constant k_2 and t_0 such that for all $t \ge t_0$,

$$\mathbb{P}\left((\mathcal{E}_3(t))^c\right) \le K\left(\frac{1}{t^7} + \frac{k_2+1}{t^3}\right).$$

Based on Lemma 5, Lemmas 3 and 4 can be proven through the following outline. The bound on the busy period in Lemma 3 relies on the property that $\mathbb{E}\left(A_i(\tau) \mid \hat{p}_i(\tau)\right) \le \mu_i - \frac{r_i}{2}$ in the event $\mathcal{E}_{32}(t)(t)$, which leads to a negative drift in the queue length. For the bound on the estimation error in Lemma 4, the property that $\mathbb{E}\left(A_i(\tau) \mid \hat{p}_i(\tau)\right) \ge \frac{\lambda p_i^*}{2}$ in the event $\mathcal{E}_{32}(t)$ guarantees that the expected number of jobs we dispatch to each server in the optimal support set is at least linear in time, resulting in enough samples for estimating the service rates of these servers. For servers outside of the optimal support set, event $\mathcal{E}_{31}(t)$ ensures that we do not dispatch exploitation jobs to those servers.

6 SIMULATION RESULTS

In this section we compare the expected regret of our proposed $K \ln t/t$-exploration policy with three other policies that are also based on multi-armed bandits: (i) an ϵ_t-exploration with a faster decaying exploration probability $\epsilon_t = K/t$, (ii) a variant of the upper confidence bound (UCB) policy [2], and (iii) a variant of Thompson sampling [1, 24], described in more detail below. Our simulation set-up consists of a system of 6 servers with service rates μ_i such that $\mu_i = 2^{i-1}\mu_1$, and $\sum_{i=1}^{6} \mu_i = 0.99$. We consider 5 different job arrival rates $\lambda = 0.1, 0.2, 0.4, 0.5$ and 0.7. To compute the regret, we find the cumulative queue length $\sum_{\tau=1}^{t}\sum_{i=1}^{K} Q_i(\tau)$ for $t \in [0, 2\times 10^7]$ for each of the policies and the optimal weighted random routing policy. The regret $\Psi_{\mathcal{P}\mathcal{P}^*}(t) = \sum_{\tau=1}^{t}\sum_{i=1}^{K}(Q_i(\tau) - Q_i^*(\tau))$ is then averaged over 20000 simulation runs. We compare the regret of our proposed policy with that of the three other policies in Fig. 5.

K/t-exploration: Instead of the $\epsilon_t = K \ln t/t$ probability of exploration used in our proposed policy, this policy sets $\epsilon_t = K/t$, which decays much faster. Because of the aggressive exploitation, this policy can exclude servers from the optimal support set, similar to the situation described in Section 4.1. As a result, we observe linearly increasing regret in Fig. 5 and it is clearly outperformed by our proposed $K \ln t/t$-exploration policy, especially for small λ. For larger λ, only a small amount of exploration is required to ensure that none of the servers in the optimal support set is excluded and hence the performance of the policy improves.

Upper Confidence Bound (UCB) variant: This is a variant of the UCB policy [2], where in each time slot, we compute the routing probability vector $f(\lambda, \boldsymbol{\mu}^{UCB}(t))$ using optimistic estimates of the

service rates $\mu_i^{UCB}(t) = \hat{\mu}_i(t) + \frac{1}{\sqrt{N_i(t)}}$, where $N_i(t)$ is the number of jobs that have departed from server i till time t. Using optimistic service rate estimates induces more exploration of slower servers by including them in the support set more often. As a result, UCB explores more aggressively than our proposed $K \ln t/t$-exploration policy and therefore, UCB performs well for small λ. However, as λ increases, the additional exploration results in a higher regret.

Thompson Sampling (TS) variant: This is a variant of the Thompson sampling [1, 24]. At each time slot, we compute the routing probability vector $f(\lambda, \boldsymbol{\mu}^{TS}(t))$ by sampling the service rates $\mu_i^{TS}(t)$ from a Beta distribution with parameters $\hat{\mu}_i(t)N_i(t)+1$ and $(1-\hat{\mu}_i(t))N_i(t)+1$. The variance of the Beta distribution is roughly $O(1/N_i(t))$. The exploration in this policy comes from the fact $\mu_i^{TS}(t)$ lies within $\hat{\mu}_i(t) \pm O(1/\sqrt{N_i(t)})$ region. While similar to UCB, TS performs less exploration of slow servers because $\mu_i^{TS}(t)$ can be lower than the optimistic estimates $\hat{\mu}_i(t)$. Thus, we observe in Fig. 5 that the regret of TS is similar to, but better than UCB.

A common trend in these results is that we need more exploration in the low λ regime and less exploration for larger λ. In the low λ regime, the optimal weighted random routing usually sends the job to the fastest server, which essentially reduces to a typical MAB setting. Hence, UCB and TS perform well in very low λ regime. However, their performance worsens as λ increases due to over-exploration. Unlike traditional MAB problems where the user either explores or exploits at each time, in queueing bandits every exploitation also acts as an exploration. As long as the servers in the optimal support set has a non-zero probability of assignment associated with it, there would be a steady flow of jobs to those servers which in turn will improve their service rate estimates.

7 CONCLUDING REMARKS

In this paper, we study the problem of job dispatching policies in a system with unknown service rates and unknown queue length information. We propose a bandit-based $K \ln t/t$-exploration policy, which uses online estimate of the service rates to dispatch jobs, and asymptotically converges to the optimal weighted random (OWR) routing policy. We characterize the finite-time regret of this policy and present simulation results to demonstrate that it performs well in all load regimes.

There are substantial open directions for future work. An immediate open challenge is to prove a matching lower bound on the regret. Unlike typical bandit problem where every wrong decision incurs a penalty, characterization of this penalty is difficult in our queueing setting. Another open direction is extending the work to characterize of regret for classes of policies that has access to the queue length information like JSQ, SED etc. The analysis of these policies is far more complicated than random routing policies, where the difference in the queue length can be characterized by the difference in the routing probabilities.

ACKNOWLEDGMENTS

The authors thank Osman Yağan for insightful initial discussions. This work was supported in part by the CMU Dean's fellowship, NSF CCF grant #2007834, NSF CNS grant #2007733, NSF CMMI Grant #1826320, NSF CNS Grant #1910112. ONR Grant N00014-19-1-2566 and a Carnegie Bosch Institute Research Award.

REFERENCES

[1] Shipra Agrawal and Navin Goyal. 2012. Analysis of Thompson Sampling for the Multi-armed Bandit Problem. In *Proc. Conf. Learning Theory (COLT)*, Vol. 23. Edinburgh, Scotland, 39.1–39.26.
[2] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-Time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47, 2–3 (May 2002), 235–256.
[3] Sayed A. Banawan and Nidal M. Zeidat. 1992. A Comparative Study of Load Sharing in Heterogeneous Multicomputer Systems. In *Proc. of the 25th Annu. Symp. on Simul.* Washington, DC, USA, 22–31.
[4] Sébastien Bubeck and Nicolò Cesa-Bianchi. 2012. *Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems.* Vol. 5. Foundations and Trends® in Machine Learning. 1–122 pages.
[5] Semih Cayci and Atilla Eryilmaz. 2017. Learning for serving deadline-constrained traffic in multi-channel wireless networks. In *Proc. Int. Symp. Modelling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, Vol. 27. Paris, France, 1–8.
[6] Tuhinangshu Choudhury, Gauri Joshi, Weina Weng, and Sanjay Shakkottai. 2021. Job Dispatching Policies for Queueing Systems with Unknown Service Rates (Extended). *arXiv:2106.04707 [eess.SY]* (June 2021).
[7] Jeffrey Dean and Luiz André Barroso. 2013. The Tail at Scale. *ACM Commun.* 56, 2 (Feb. 2013), 74–80.
[8] Atilla Eryilmaz and R. Srikant. 2012. Asymptotically Tight Steady-State Queue Length Bounds Implied by Drift Conditions. *Queueing Syst.* 72, 3–4 (Dec. 2012), 311–359.
[9] Santosh Fatale, Kavya Bhandari, Urvidh Narula, Sharayu Moharir, , and Manjesh Hanawal. 2020. Regret of Age-of-Information Bandits. *arXiv:2001.09317 [eess.SY]* (June 2020).
[10] GJ Foschini. 1977. On heavy traffic diffusion analysis and dynamic routing in packet switched networks. *Computer Performance* 10 (1977), 499–513.
[11] Mor Harchol-Balter. 2013. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action* (1st ed.). Cambridge University Press.
[12] G.P. Klimov. 1974. Time-sharing service systems I. *Theory Prob. Appl.* 19 (1974), 532 –– 551.
[13] Subhashini Krishnasamy, Ari Arapostathis, Ramesh Johari, and Sanjay Shakkottai. 2018. On Learning the cμ Rule in Single and Parallel Server Networks. In *Proc. Ann. Allerton Conf. Communication, Control and Computing*. IEEE, Monticello, IL, USA, 153–154.
[14] Subhashini Krishnasamy, Rajat Sen, Ramesh Johari, and Sanjay Shakkottai. 2016. Regret of Queueing Bandits. In *Advances Neural Information Processing Systems (NEURIPS)*, Vol. 29.
[15] T.L Lai and Herbert Robbins. 1985. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.* 6, 1 (1985), 4–22.
[16] Tor Lattimore and Csaba Szepesvári. 2020. *Bandit Algorithms.* Cambridge University Press.
[17] Xin Liu, Bin Li, Pengyi Shi, and Lei Ying. 2020. POND: Pessimistic-Optimistic oNline Dispatch. *arXiv:2010.09995 [cs.LG]* (Oct. 2020).
[18] Aditya Mahajan and Demosthenis Teneketzis. 2008. Multi-armed bandit problems. *Foundations and applications of sensor management* (2008), 121 – 151.
[19] Michael Mitzenmacher. 1996. Load balancing and density dependent jump Markov processes. In *Proce. Conf. Found. of Comput. Sci.* IEEE, Burlington, VT, USA, 213–222.
[20] Michael David Mitzenmacher. 2001. The power of two choices in randomized load balancing. *IEEE Trans. Parallel Distrib. Syst.* 12, 10 (2001), 1094–1104.
[21] Sheldon M. Ross and Erol A. Peköz. 2007. *A second course in probability.* www.ProbabilityBookstore.com.
[22] R. Srikant and Lei Ying. 2014. *Communication Networks: An Optimization, Control and Stochastic Networks Perspective.* Cambridge University Press, USA.
[23] Thomas Stahlbuhk, Brooke Shrader, and Eytan Modiano. 2018. Learning Algorithms for Minimizing Queue Length Regret. In *Proc. IEEE Int. Symp. Information Theory (ISIT)*. 1001–1005.
[24] William R. Thompson. 1933. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. *Biometrika* 25, 3/4 (1933), 285–294.
[25] Nikita Dmitrievna Vvedenskaya, Roland L'vovich Dobrushin, and Fridrikh Izrailevich Karpelevich. 1996. Queueing system with selection of the shortest of two queues: An asymptotic approach. *Problemy Peredachi Informatsii* 32, 1 (1996), 20–34.
[26] N.S. Walton. 2014. Two queues with non-stochastic arrivals. *Operations Research Letters* 42, 1 (2014), 53 – 57.
[27] Richard R. Weber. 1978. On the optimal assignment of customers to parallel servers. *J. Appl. Probab.* 15, 2 (1978), 406–413.
[28] Wentao Weng, Xingyu Zhou, and R. Srikant. 2020. Optimal Load Balancing in Bipartite Graphs. *arXiv:2008.08830 [cs.PF]* (Aug. 2020).