Integrating Online Learning and Adaptive Control in Queueing Systems with Uncertain Payoffs

Wei-Kang Hsu*, Jiaming Xu†, Xiaojun Lin* and Mark R. Bell*
*School of Electrical and Computer Engineering, Purdue University, West Lafayette
†Krannert School of Management, Purdue University, West Lafayette
Email: {wkhsu, xu972}@purdue.edu, {linx, mrb}@ecn.purdue.edu

***Abstract*—We study task assignment in online service platforms where un-labeled clients arrive according to a stochastic process and each client brings a random number of tasks. As tasks are assigned to servers, they produce client/server-dependent random payoffs. The goal of the system operator is to maximize the expected payoff per unit time subject to the servers' capacity constraints. However, both the statistics of the dynamic client population and the client-specific payoff vectors are unknown to the operator. Thus, the operator must design task-assignment policies that integrate adaptive control (of the queueing system) with online learning (of the clients' payoff vectors). A key challenge in such integration is how to account for the non-trivial closed-loop interactions between the queueing process and the learning process, which may significantly degrade system performance. We propose a new utility-guided online learning and task assignment algorithm that seamlessly integrates learning with control to address such difficulty. Our analysis shows that, compared to an oracle that knows all client dynamics and payoff vectors beforehand, the gap of the expected payoff per unit time of our proposed algorithm in a finite T horizon is bounded by $\beta_1/V + \beta_2\sqrt{\log N/N} + \beta_3 N(V+1)/T$, where V is a tuning parameter of the algorithm, and $\beta_1, \beta_2, \beta_3$ only depend on arrival/service rates and the number of client classes/servers. Through simulations, we show that our proposed algorithm significantly outperforms a myopic matching policy and a standard queue-length based policy that does not explicitly address the closed-loop interactions between queueing and learning.**

I. Introduction

Driven by advances in communication and computing, new generations of online service platforms have transformed business models in many domains, from online labor market of freelance work, online hotel rental, online education, crowd-sourcing, to online advertising and many more. By bringing unprecedented numbers of clients and service providers together, these online service platforms greatly increase access to service, lower the barrier-to-entry for competition, improve resource utilization, reduce cost and delay, and thus enhance the overall well-being of society and our daily life.

One of the key control decisions in operating such online service platforms is how to assign clients to servers to attain the maximum system benefit. However, such decisions are challenging because there often exists significant uncertainty in both client payoffs and client dynamics. First, there is significant uncertainty in the quality, i.e., payoff, of a particular assignment between a client (needing service) and a server (providing service). For example, in online ad platforms, when an ad from a particular ad campaign is displayed in response to a search request for a given keyword, the click-through rate is unknown in advance [1]. Similarly, in crowd-sourcing, the efficiency of completing a particular task by a given worker is unknown *a priori* [2]. The online service platforms have to learn the payoff parameters from the payoff feedback of past assignments. Second, the population of clients is often highly dynamic. For example, ad campaigns arrive and depart constantly in online ad platforms [3]; so do requestors in crowd-sourcing. The statistics of such arrivals and departures are often unknown beforehand, resulting in uncertain queueing dynamics. Further, when a new client arrives, it brings a new set of payoff parameters, which may need to be learned from scratch. Therefore, the operator of these platforms must not only continuously learn the payoffs associated with each new client, but also adaptively control the assignment and resource allocation in response to the uncertain arrival/departure dynamics. Thus, it is imperative to study both online learning (of uncertain payoffs) and adaptive control (of uncertain queueing dynamics) in a unified framework to achieve optimal performance in such a dynamic and uncertain environment.

Motivated by these features, in this paper we focus on the following queueing system model that incorporates uncertainty in both client dynamics and payoffs. Note that on many online service platforms, the operator often has enough aggregate information on the servers' features, and servers arrive at and depart from the system at a slower time scale than clients [4], [5], [6]. Thus, in this work we assume that there are a fixed number of servers. Clients of multiple classes arrive at the system with unknown arrival rates. Using the earlier examples, a client may represent an ad campaign or a crowd-sourcing requestor. Each client brings a random number of tasks, which may correspond to the ads from the same ad campaign or the crowd-sourcing tasks of the same requestor. Associated with each class, there is a payoff vector over all servers. As the tasks of a client are assigned to different servers, they produce random payoff feedback with mean given by the underlying class-dependent payoff vector. *However, neither the class label of a client nor its payoff vector is known to the operator.* Thus, the operator has to learn the payoff vector of a new client from the random payoff feedback, and then decide how to match clients with servers.

We note that the aforementioned model differs significantly from both classical online learning problems and adaptive control problems in the literature. On the one hand, while online learning has often been studied as a MAB (multi-armed bandits) problem, most existing studies ignore the client dynamics and focus on either one client [7], [8], [9], [10] or a fixed set of clients [11], [12], [13], [14], [15]. Our model is also different from open-bandit processes (where arms are dynamic and follow a Markov chain with known statistics [16]), and contextual bandits (which assume known label, i.e., context, for each incoming client [17]). On the other hand, although adaptive control and online matching policies under uncertain dynamics have been provided, e.g., for online ad [3], [18] and/or queueing networks in general [19], [20], [21], [22], these studies usually assume that the payoff vectors of the clients are known.

The main contribution of this paper is therefore to develop new schemes that integrate online learning with adaptive control to address uncertainty in both payoff parameters and client dynamics. We propose a new utility-guided online learning and task assignment algorithm that can achieve near-optimal system payoff even compared to an "oracle" that knows the statistics of client dynamics and all clients' payoff vectors beforehand. Specifically, we show that the gap between the expected payoff per unit time achieved by our proposed algorithm and that achieved by the "oracle" in a finite T horizon is at most the sum of three terms: the first one is of the order $1/V$ where V is a parameter of the proposed algorithm that can be taken to a large value; the second one is of the order $\sqrt{\log N/N}$ where N is the average number of tasks per client; the third one is of the order $N(V+1)/T$ which decreases as the time T increases. These three terms have the following natural physical interpretations. The first $1/V$ term is due to the uncertainty in client arrival process; the second $\sqrt{\log N/N}$ term is because of learning from noisy payoff feedback; the third $N(V+1)/T$ term characterizes the payoff loss incurred by the backlogged clients.

Our proof builds upon the standard Lyapunov drift technique [20], [22], [23], [24] and finite-time regret analysis [8]; however, we make two major innovations, which could be of independent interest. First, to obtain $\sqrt{\log N/N}$ loss in payoff learning, we combine a martingale argument and a comparison argument to capture both the impact of the payoff estimation errors on the queueing dynamics, and the impact of congestion on the rate of payoff learning. Second, to derive the NV/T loss in finite time horizon, we upper bound the mean number of backlogged clients in the system by establishing a coupling between the current system and a Geom/Geom/m queue.

There are a few recent studies on integration of learning and queueing [4], [5], [6], [25], [26]. Compared to these related studies, our work is significantly different in the following aspects. First of all, while the previous work [4], [6] also investigates payoff loss in queueing systems, its analysis focuses exclusively on the stationary setting. In contrast, our work carefully characterizes the transient behavior of the system and obtains a payoff gap which holds for any finite time horizon, providing important design insight when real systems either operate at early phases or experience recent changes of

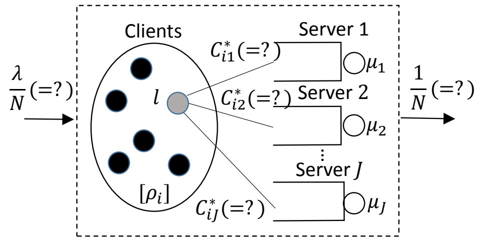

Fig. 1. Uncertain client dynamics and payoffs.

statistics. Moreover, policies in [4], [5], [6] explicitly divide exploration (where learning occurs) and exploitation (where queueing and control occur) into two stages, by assuming perfect knowledge of various uncertainty, such as class-dependent payoff vectors, the number of tasks per client, and/or "shadow prices" (which in turn require knowledge of the statistics of the client arrival dynamics). On the contrary, our algorithm requires no such prior knowledge and seamlessly integrates exploration, exploitation, and dynamic assignment at all times. Finally, our proposed algorithm can scale to large systems with many clients and servers, which is in sharp contrast to the previous work in both [25] (which only deals with two types of clients) and [26] (which uses an exponential number of virtual queues).

The rest of the paper is organized as follows. The system model is defined in Section II. Our proposed algorithm and main analytical results are presented in Section III. Simulation results are shown in Section IV, and key proof steps are sketched in Section V. Finally, we conclude and discuss possible future directions.

II. System Model

We model an online service platform as a multi-server queueing system that can process tasks from a dynamic population of clients, as shown in Fig. 1. Time is slotted and the system is empty at time 0. At the beginning of each time-slot $t \geq 1$, a new client arrives with probability λ/N, independently of other time-slots. Upon arrival, the client carries a random number of tasks that is geometrically distributed with mean N. Note that in this way the total rate of arriving tasks is λ. A client leaves the system once all of her tasks have been served. We assume that each client must belong to one of I classes. There is a probability distribution $[\rho_i, i = 1, ..., I]$ with $\sum_{i=1}^{I} \rho_i = 1$, such that a newly-arriving client belongs to class i with probability ρ_i, independently of other clients. Let $\mathcal{S} = \{1, 2, \ldots, J\}$ denote the fixed set of servers. Each server $j \in \mathcal{S}$ can serve exactly μ_j tasks in a time-slot. Let $\mu = \sum_{j=1}^{J} \mu_j$.

Each class i is associated with a payoff vector $[C_{ij}^*, j = 1, ..., J]$, where $C_{ij}^* \in [0, 1]$ is the expected payoff when a task from a class-i client is served by server j. Note that all tasks associated with a client have the same payoff vector. *However, the arrival rate λ, the class distribution ρ_i, the expected number N of tasks per client, the class label of a new client, her total number of tasks, and her payoff vector are* *all unknown to the system.* The system does know the identity of the servers and their service rates μ_j. Another important quantity that the system can learn from is that, after a task from a client is served by server j, the system can observe a noisy payoff of the task. Conditioned on the task being from a class-i client, this noisy payoff is assumed to be a Bernoulli random variable with mean C^*_{ij}, (conditionally) independently of other tasks and servers.

The goal of the system is then to use the noisy payoff feedback to learn the payoff vectors of the clients and to assign their tasks to servers in order to maximize the total system payoff. Such decisions must be adaptively made without knowing the statistics of the client arrivals and departures. At each time-slot t, let $n(t)$ be the total number of clients in the system (including any new arrival at the beginning of the time-slot). (Note that $n(0) = 0$.) Let $p^l_j(t)$ be the expected number of tasks from the l-th client that are assigned to server j in this time-slot, $l = 1, ..., n(t)$. Then, the decision at time t of a policy Π maps from the current state of the system (including all payoff feedback observed before time t) to the quantities $p^l_j(t)$. Such decisions lead to the evolution of the following two types of queues. First, let $q_j(t)$ denote the number of tasks waiting to be served at server j at the beginning of time-slot t. Let $Y^l_j(t)$ be the actual number of tasks from the l-th client that are assigned to server j at time-slot t, with mean given by $p^l_j(t)$. The dynamics of the task-queue $q_j(t)$ can then be described as

$$q_j(t+1) = \max\left[q_j(t) + \sum_{l=1}^{n(t)} Y^l_j(t) - \mu_j, 0\right]. \quad (1)$$

Second, let $n_i(t)$ be the number of class-i clients in the system at the beginning of time t. (We caution that, while the operator sees $n(t)$, $n_i(t)$ cannot be directly observed because the class labels of the clients are unknown.) The dynamics of the client-queue $n_i(t)$ can be described as

$$n_i(t+1) = n_i(t) + U_i(t+1) - D_i(t), \quad (2)$$

where $U_i(t+1)$ is the number of client arrivals of class i at the beginning of time $t+1$ and is Bernoulli with mean λ_i/N, and $D_i(t)$ is the number of client departures of class i at time-slot t. Because the total number of tasks per client is assumed to be geometrically distributed, conditional on $[Y^l_j(t)]$ and $[n_i(t), i \in I]$, the departure $[D_i(t), i \in I]$ is independent of everything else.

The expected payoff per unit time of such a policy Π for a given time horizon T is defined as:

$$R_T(\Pi) = \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{J}\mathbb{E}\left[\sum_{l=1}^{n(t)} p^l_j(t) C^*_{i(l),j}\right], \quad (3)$$

where $i(l)$ denotes the underlying (but unknown) class of the l-th client. In contrast, if an "oracle" knew in advance the statistics of the arrival and departure dynamics (i.e., λ, ρ_i and N), as well as the class label and payoff vector of each client, one can formulate the following linear program:

$$R^* = \max_{[p_{ij}]\geq 0} \quad \lambda \sum_{i=1}^{I}\rho_i \sum_{j=1}^{J} p_{ij} C^*_{ij} \quad (4)$$

$$\text{subject to} \quad \lambda \sum_{i=1}^{I} \rho_i p_{ij} \leq \mu_j \text{ for all servers } j, \quad (5)$$

$$\sum_{j=1}^{J} p_{ij} = 1 \text{ for all classes } i, \quad (6)$$

where p_{ij} is the probability that a task from a class-i client is assigned to server j. It is not difficult to show that R^* provides an upper bound for the expected payoff per unit time $R_T(\Pi)$ under any policy Π [4], [6]. Thus, our goal is to achieve a provably small gap $R^* - R_T(\Pi)$ for any given time T.

III. Dynamic Assignment with Online Learning under Uncertainties

A. Difficulty of Classical Queue-length Based Control

A classical approach to adaptive control in the presence of uncertain arrival/departure dynamics is to use queue-length based policies [19], [20], [27], [28], [21]. This line of work assumes perfect knowledge of the payoff vector for each client. By building up queues of unserved tasks at the servers, the system operator uses the queue length q_j at server j as a dynamic "shadow price" to capture the congestion level at the server. The system operator then adjusts each client's payoff parameter C^*_{ij} to $C^*_{ij} - q_j/V$ with a proper choice of V, and assigns the next task to the server with the highest adjusted payoff. When the payoff vector of each client is known in advance, this type of queue-length based control can be shown to attain near-optimal system payoff when the parameter V is large [3].

However, when the payoff vector of each client is unknown, such queue-length based control leads to complicated closed-loop interactions between queueing and learning. In one direction, excessive *queueing* degrades *learning* performance in at least two ways. First, because the operator is only able to receive the noisy payoff feedback of a task *after* the task is served, excessive task-queues at server-sides (esp. when V is large) not only delay the service of the tasks, but also delay the payoff feedback. We refer to this effect as the *payoff-feedback delay*. Second, because all clients compete for the limited server capacity, the learning process of some clients will inevitably be slowed down, which also results in poor estimate of their payoff vectors. We refer to this second effect as the *learning slow-down*. In the opposite direction, ineffective *learning* in turn affects *queueing* because assignment decisions have to be made based on delayed or inaccurate payoff estimates. Such sub-optimal decisions not only lower system payoff, but also increase queueing delay. Together, the above closed-loop interactions between queueing and learning will produce complex system dynamics which, if not designed and controlled properly, can severely degrade system performance.

In this section, we present a new algorithm that seamlessly integrates online learning with dynamic task assignment to address the aforementioned closed-loop interactions. Unlike classical queue-length based control policies that rely on the task-queues as the "shadow prices," we instead eliminate the task-queues at the servers, and thus eliminate the "payoff-feedback delay" altogether. This is achieved by controlling the number of tasks assigned to each server j at each time-slot to be always no greater than its service rate μ_j. Therefore, the length of the server-side task-queue $q_j(t)$ given by (1) is trivially zero at all times. However, without the server-side task-queues or "shadow prices", we need another congestion indicator to guide us in the dynamic assignment of tasks. Here, we propose to rely on the number of backlogged clients in the system. Specifically, we use the solution to a utility-maximization problem (that is based on the current number of backlogged clients in the system) to help us trade-off between the current and future payoffs. The parameters of the utility-maximization problem are carefully chosen to also control the *learning slow-down* of all clients in a fair manner. We remark that while the structure of our proposed utility-guided algorithm bears some similarity to that of flow-level congestion control in communication networks [22], [23], [24], the focus therein was only on the system stability (but not the system payoff), and there was no online learning of uncertain parameters. *To the best of our knowledge, our work is the first in the literature that uses utility-guided adaptive control policies with online learning to optimize system payoff.*

B. Utility-Guided Dynamic Assignment with Online Learning

We present our algorithm in Algorithm 1. At each time-slot t, the algorithm operates in three steps. Step 1 generates an estimate $[C_j^l(t)]$ of the payoff vector for each client l based on previous payoff feedback (Lines 4-7). Note that $n(t)$ is the current number of clients in the system, including any newly-arriving client. For each client $l = 1, ..., n(t)$, if no task of this client has been assigned to server j yet, we set $C_j^l(t) = 1$ (Line 6); otherwise, we use a truncated Upper-Confidence-Bound (UCB) estimate [8] to generate $C_j^l(t)$ (Line 7):

$$C_j^l(t) = \min\left\{\overline{C}_j^l(t-1) + \sqrt{\frac{2\log h^l(t-1)}{h_j^l(t-1)}},\ 1\right\}, \quad (9)$$

where $h_j^l(t-1)$ is the number of tasks from client l that have previously been assigned to server j before the end of the $(t-1)$-th time-slot and $h^l(t-1) = \sum_{j=1}^J h_j^l(t-1)$; $\overline{C}_j^l(t-1)$ is the empirical average payoff of client l based on the received noisy payoff feedback for server j until the end of the $(t-1)$-th time-slot. Since the true payoffs C_{ij}^*'s are within $[0,1]$, we truncate the UCB estimate at threshold 1 in (9). Then, Step 2 solves a maximization problem (Line 9), subject to the capacity constraint of each server, to obtain the values of $p_j^l(t)$, which corresponds to the expected number of new tasks of client l to be assigned to server j in time-slot t. The objective (7) can be viewed as the sum of some utility functions over all clients. The parameter γ in (7) is chosen to be strictly larger than 1, and V is a positive parameter. Finally, Step 3 determines the exact number $Y_j^l(t)$ of tasks from client l that are assigned to server j (Lines 12-14). The values of $Y_j^l(t)$ are randomly chosen in such a way that (i) each server j receives at most μ_j tasks, i.e., $\sum_{l=1}^{n(t)} Y_j^l(t) \le \mu_j$; and (ii) the expected value of $Y_j^l(t)$ is equal to $p_j^l(t)$. The tasks are then sent to the servers and new noisy payoffs are received (Lines 15-19).

Algorithm 1: Utility-Guided Dynamic Assignment with Online Learning

```
1  For every time slot $t$:
2  Update the total number of clients $n(t)$ (including newly arriving clients)
3  Step 1: Form truncated UCB payoff estimates
4  for $l = 1 : n(t)$ do
5      for $j = 1 : J$ do
6          if $h_j^l(t-1) = 0$ or client $l$ is new then $C_j^l(t) \leftarrow 1$; $h_j^l(t) \leftarrow 0$;
7          else Set $C_j^l(t)$ according to (9) ;
8  Step 2: Solve $[p_j^l(t)]$ for the optimization problem
9
   $\max_{[p_j^l]\ge 0} \sum_{l=1}^{n(t)} \left\{ \frac{1}{V}\log\left(\sum_{j=1}^J p_j^l\right) + \sum_{j=1}^J p_j^l (C_j^l(t) - \gamma) \right\}$   (7)
   sub to $\sum_{l=1}^{n(t)} p_j^l \le \mu_j$, for all servers $j \in \mathcal{S}$.   (8)
10 Step 3: Assign tasks and obtain noisy payoff feedback
11 Initialize to zero the number of tasks from client $l$ to be assigned to server $j$, i.e., $Y_j^l(t) = 0$.
12 for $j = 1 : J$ do
13     for $\nu = 1 : \mu_j$ do
14         Choose a client $l^*$ randomly such that the probability of choosing client $l$ is equal to $p_j^l(t)/\mu_j$. Assign one task from client $l^*$ to server $j$ and let $Y_j^{l^*}(t) \leftarrow Y_j^{l^*}(t) + 1$ ;
15 for $l = 1 : n(t)$ do
16     for $j = 1 : J$ do
17         Observe $Y_j^l(t)$ number of Bernoulli noisy payoffs $X_j^l(t,1), \dots, X_j^l(t, Y_j^l(t)) \overset{\text{i.i.d.}}{\sim} \text{Bern}\left(C_{i(l),j}^*\right)$;
18         Update $h_j^l(t)$ and $\overline{C}_j^l(t)$ according to $Y_j^l(t)$ and $X_j^l(t,\cdot)$;
19     $h^l(t) = \sum_{j=1}^J h_j^l(t)$ ;
20 Clients with no remaining tasks leave the system.
```

Before we present our analytical results, we make several remarks on the design of our proposed algorithm.

Seamless Integration of Learning and Control: First, recall that the policies in [4], [5], [6] separate exploration and exploitation into distinct stages by assuming perfect knowledge of class-dependent payoff vectors, the total number of tasks per client, and/or "shadow prices." In contrast, our algorithm does not require such prior knowledge at all and seamlessly integrates exploration, exploitation, and dynamic assignment at all times.

Zero Payoff-Feedback Delay: Second, note that by Step 3 of the algorithm, the number of assigned tasks to each server j is no more than the service rate μ_j. Since server j can serve exactly μ_j tasks per time-slot, there is no longer any "payoff-feedback delay," i.e., all payoff feedback will be immediately revealed at the end of the time-slot.

The Importance of Fairness: Third, if we removed the logarithmic term in (7) and set $\gamma = 0$, then the maximization problem in (7) would have become a myopic matching policy that maximizes the total payoff in time-slot t based on the current payoff estimates. However, such a myopic matching policy focuses too much on the current payoff, and as a result underperforms in terms of the long-term average payoff. Instead, the logarithmic term in (7) serves as a concave utility function that promotes fairness [22], [23], [24], so that even clients with low payoff estimates can still receive some service (i.e., $\sum_{j=1}^{J} p_j^l(t)$ is always strictly positive). This fairness property is desirable in two ways. First, it has an eye for the future (which is somewhat related to the fact that fair congestion-control ensures long-term system stability [24]), so that we can strike the right balance between current and future payoffs. Second, it also controls the *learning slow-down* of all clients in a fair manner. Specifically, thanks to this fairness property, we can show that the "learning rate" of each client based on imprecise payoff estimates will not be too far off from the "learning rate" of the client based on its true payoff vector, which constitutes a crucial step in our analysis (see Section V-C).

Parameter Choice: Fourth, the choice $\gamma > 1$ in (7) and the truncation of the UCB estimates in (9) also play a crucial role in achieving near-optimal system payoff. If $\gamma = 0$, then every client l will have the incentive to increase $\sum_{j=1}^{J} p_j^l(t)$ by utilizing as many available servers as possible, *even when the server payoff is very close to zero*. Such an aggressive approach typically leads to suboptimal performance because in many settings the optimal policy must restrict some clients to only use the high-payoff servers. Instead, by choosing $\gamma > 1 \geq C_j^l(t)$, the second term inside the summation in (7) becomes negative. As a result, the operator assigns clients to low-payoff servers only if the derivative of the logarithmic term, which is equal to $1/[V \sum_{j=1}^{J} p_j^l(t)]$, is sufficiently large. This feature leads to an inherent "conservativeness" of our algorithm in choosing low-payoff servers. As the parameter V increases, our algorithm becomes more and more conservative in choosing low-payoff servers, which benefits the long-term payoff. On the other hand, it also increases the number of clients backlogged in the system, leading to extra payoff loss for a finite time horizon T. Our theoretical results below will capture this tradeoff.

Complexity: Last but not least, the maximization problem in Step 2 is a convex program and can be effectively solved. Thus, our proposed algorithm can scale to large systems with many clients and servers, which is in contrast to the work in both [25] (which only deals with two types of clients) and [26] (which uses an exponential number of virtual queues).

C. Main Results

We now present our main results for the performance guarantees of the proposed algorithm. For simplicity, we denote $\lambda_i = \lambda \rho_i$, and hence $\lambda = \sum_{i=1}^{I} \lambda_i$. Recall that $\mu = \sum_{j=1}^{J} \mu_j$ and $n(0) = \sum_{i=1}^{I} n_i(0) = 0$. The first results gives an upper bound to the mean number of backlogged clients in the system at any time t.

Theorem 1. *Suppose that the arrival rate λ is strictly less than the total service capacity, i.e., $\lambda < \mu$. Then for any time t,*

$$\mathbb{E}\left[n(t)\right] \leq \frac{2\mu}{\mu - \lambda}\left(1 + \frac{\mu^2 \gamma}{\gamma - 1}\right) + \mu \gamma V. \tag{10}$$

Theorem 1 immediately implies the system is stable in the mean [29]. The upper bound in (10) characterizes the effect of V. As V increases, our algorithm becomes more conservative in choosing low-payoff servers, leading to more clients backlogged in the system.

Departing from the standard Lyapunov technique in proving system stability [22], [23], [24], our proof of Theorem 1 relies on a careful coupling between $n(t)$ and a Geom/Geom/m queue with Bernoulli arrivals and Binomial departures (see [30]).

The second result below further characterizes the payoff gap of the proposed algorithm compared to the oracle.

Theorem 2. *Suppose $N \log N \geq 1$. The gap between the upper bound* (4) *and the expected payoff of Algorithm 1 at any time horizon T is bounded by:*

$$\sum_{i=1}^{I}\sum_{j=1}^{J} \lambda_i p_{ij}^* C_{ij}^* - \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{J} \mathbb{E}\left[\sum_{l=1}^{n(t)} p_j^l(t) C_{i(l),j}^*\right] \leq \frac{\beta_1}{V} + \beta_2 \sqrt{\frac{\log N}{N}} + \beta_3 \frac{N(V+1)}{T}, \tag{11}$$

where $[p_{ij}^*]$ is the optimal solution to (4)*, and*

$$\begin{aligned}
\beta_1 &= \frac{I}{2} + \frac{\mu^2}{2}\left(1 + \frac{\gamma^2}{(\gamma-1)^2 N}\right)\sum_{i=1}^{I}\frac{1}{\lambda_i}, \\
\beta_2 &= 4\sqrt{2}\lambda(J+\mu) + 3\lambda\left(1 + \frac{J\gamma^2}{(1-\gamma)^2}\right)\mu, \\
\beta_3 &= \frac{2\mu\gamma}{\mu-\lambda} + \frac{2\mu^3\gamma^2}{(\mu-\lambda)(\gamma-1)} + \mu\gamma^2.
\end{aligned} \tag{12}$$

To the best of our knowledge, this is the first result in the literature that characterizes the payoff gap for this type of utility-guided adaptive controllers under payoff uncertainty. The above result is quite appealing as it separately captures the impact of the uncertainty in client dynamics and the impact

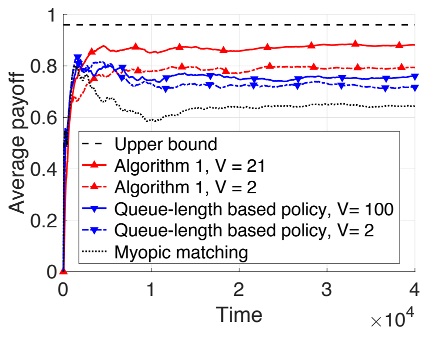

Fig. 2. The evolution of the time-averaged payoff under the three policies as simulation time advances. The upper bound (4) is 0.96 (the horizontal line).

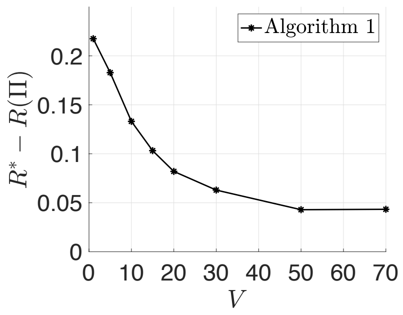

Fig. 3. With a fixed $N = 100$, the payoff gap of Algorithm 1 decreases and eventually saturates as V increases.

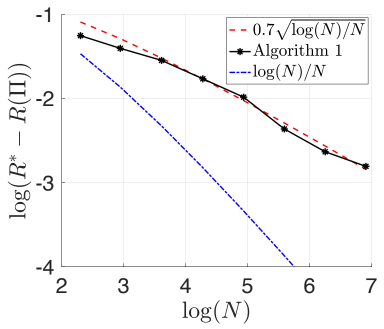

Fig. 4. With $V = \sqrt{N/\log N}$, the payoff gap of Algorithm 1 scales as $\Theta(\sqrt{\log N/N})$.

of the uncertainty in payoffs for any finite time horizon T. In (11), the first term on the right-hand-side is of the order $1/V$, capturing the impact of the uncertainty in client dynamics (e.g., we do not know what are the values of λ_i and N). The second term in (11) is of the order $\sqrt{\log N/N}$ and related to the notion of "regret" in typical MAB problems. It captures the payoff loss due to the uncertainty in payoffs as a function of the total number of tasks per client, i.e, the tradeoff between exploration and exploitation [7], [8]. The third term in (11) is of the order $N(V+1)/T$, characterizing the payoff loss incurred by clients backlogged in the system. Given T, the first term and the third term reveal an interesting tradeoff as V increases. On the one hand, the first term will approach zero at the speed of $1/V$, indicating that the policy adapts to the unknown client dynamics; on the other hand, the third term will increase linearly with V due to the payoff loss incurred by clients backlogged in the system. If N and T are known in advance, we can tune V to be $\min\{T/N, \sqrt{T/N}\}$, so that the payoff-gap upper-bound is minimized and becomes of the order $\max\{N/T, \sqrt{N/T}\}+\sqrt{\log N/N}$. As a consequence, as T increases, the payoff gap first decreases at a rate of T^{-1} for $T \leq N$ and then at a rate of $T^{-1/2}$ for $N \leq T \leq N^2/\log N$ and finally gets saturated at $\sqrt{\log N/N}$ for $T \geq N^2/\log N$.

The $\sqrt{\log N/N}$ regret here may seem inferior compared to the $\log N/N$ regret in [4], [6]. However, the regret in [4], [6] only holds under the stationary setting $T \to \infty$ and assuming knowledge of various uncertainty, such as the class-dependent payoff vectors, the number of tasks per client, and/or "shadow prices" (which in turn require knowledge of the statistics of the client arrival dynamics). In contrast, our payoff gap characterizes the transient behavior of the system and holds for any finite time T and finite $N \geq 2$ without any such prior knowledge. It remains open whether our $\sqrt{\log N/N}$ regret can be further reduced to $\log N/N$.

Our proof of Theorem 2 builds upon the standard Lyapunov drift technique [20], [22], [23], [24] and finite-time regret analysis [8]; however, our proof employs new techniques to carefully account for the closed-loop interactions between the queueing dynamics and the learning processes (see Section V).

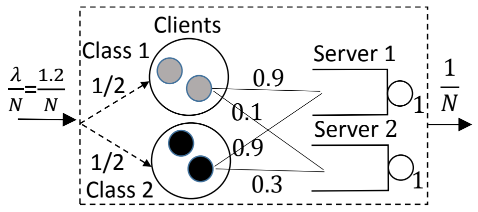

Fig. 5. The simulation setup.

IV. Numerical Results

We consider the setup as shown in Fig. 5. In particular, there are two servers and two classes of clients. Define the key parameters as follows: $\mu_j = 1$ for each server $j = 1, 2$; $\lambda_i = 0.6$ for each class $i = 1, 2$, and hence $\lambda = \lambda_1 + \lambda_2 = 1.2$. The expected number of tasks per client is initially set to be $N = 100$, but we will vary the value later. The true payoff vectors for class 1 and class 2 are given by $[0.9\ 0.1]$ and $[0.9\ 0.3]$, respectively, although they are unknown to the operator. Note that server 1 has larger expected payoff for both class 1 and class 2. However, its service rate is insufficient to support all clients. Hence, this contention must be carefully controlled when the system aims to maximize the payoff. For a given policy Π and simulation time T, we report the average system payoff $R_T(\Pi)$ per unit time.

A. Two Other Policies for Comparison

We compare our proposed Algorithm 1 with the myopic matching policy and a queue-length based policy. These two policies also use UCB to estimate unknown payoff vectors. The myopic matching policy aims at maximizing the total payoff for the current time-slot. Specifically, at each time slot t, the policy solves a modified maximization problem of the form (7)–(8), but with the logarithmic term removed and with $\gamma = 0$. Then, based on the solution to this modified maximization problem, the tasks are assigned to each server in the same way as described in Step 3 in Algorithm 1. We expect that the myopic matching policy incurs relatively large payoff loss because it does not look at the future.

The queue-length based policy maintains a queue of tasks at each server j, and uses the length of this task-queue $q_j(t)$ at time-slot t to indicate the congestion level at server j. At each time-slot t and for each client l, the operator finds the server $j^*(l)$ with the highest queue-adjusted payoff, i.e., $j^*(l) = \arg\max_j \{C_j^l(t) - q_j(t)/V\}$. The operator then adds one task from each client l to the end of the task queue at the server $j^*(l)$. Every server j then processes $\mu_j = 1$ task from the head of its own queue and observes the random payoff generated. As discussed in Section III-A, one weakness of such a queue-length based policy is that, when tasks are waiting in the queues, the system cannot observe their payoff feedback right away. Hence, there is significant payoff feedback delay, which in turn leads to suboptimal assignment decisions for subsequent tasks.

B. Performance Comparisons

We fix $\gamma = 1.1$ for our proposed Algorithm 1 in all experiements, but we may vary N and V. The first set of experiements give a general feel of the dynamics of different policies. In Fig. 2, we fix $N = 100$ and plot the evolution of the time-averaged system payoff up to time T under the three policies (with different V), as the simulation time T advances. We can see that, even when N is not very large ($N = 100$), the system payoff under our proposed Algorithm 1 with $V = 21$ (the solid curve with marker ▲) approaches the upper bound (4) (the horizontal dashed line). Further, comparing $V = 2$ (marker ▲, dashed curve) with $V = 21$ (marker ▲, solid curve), we observe significantly higher system payoff under Algorithm 1 when V increases. In comparison, the payoff achieved by the myopic matching policy (the lowest dotted curve) is significantly lower. The performance of the queue-length based policy (the two curves with marker ▼) also exhibits a noticable gap from that of the proposed Algorithm 1. Further, even when V increases from $V = 2$ (marker ▼, dashed curve) to $V = 100$ (marker ▼, solid curve), the improvement of the queue-length based policy is quite limited. This result suggests that, due to the increase in payoff-feedback delay, controlling V is not effective in improving the performance of the queue-length based policy.

In Fig. 3, we fix $N = 100$, increase V and plot the payoff gap (compared to the upper bound (4)) of our proposed Algorithm 1 over $T = 7 \times 10^5$ time slots. Each plotted data point represents the average of 5 independent runs. We can observe that initially the payoff gap decreases significantly with V, but eventually it saturates at $V \geq 50$. This is because the regret due to small N, i.e., the second term of (11), eventually dominates the payoff gap when V is large. A similar figure with increasing N but fixed V (not shown) shows a similar saturation behavior when N is large. Such saturation makes it difficult to assess whether the scaling reported in (11) is accurate. To answer this question, we next set $V = \sqrt{N/\log N}$. In this way, both the first and second terms in (11) are of the order $\Theta(\sqrt{\log N/N})$. We then increase N (and V simultaneously) and show in Fig. 4 how the payoff gap of our Algorithm 1 decreases with N on a log-log scale. The result indeed matches well with a $\Theta(\sqrt{\log N/N})$ scaling. Note that there is a noticable difference in slope from the $\Theta(\log N/N)$ scaling. The difference remains even if we set V to be larger at $V = N/\log N$. Thus, the results suggest that the performance bound in (11) for our proposed Algorithm 1 may be tight.

V. Proofs

Due to space constraint, here we sketch the proof of Theorem 2. The omitted details along with the proof of Theorem 1 are available in [30].

A. Equivalent Reformulation

We will use a Lyapunov-drift analysis with special modifications to account for the learning of uncertain payoffs. Note that for the purpose of this drift analysis, we can use the underlying class label of each client to keep track of the system dynamics, even though our algorithm does not know this underlying class label. Thus, we re-label the $n(t)$ clients at time-slot t as follows. Recall that $n_i(t)$ is the number of clients at time-slot t whose underlying class is i. Let $\mathcal{I}(t) = \{i : n_i(t) \geq 1\}$. We have $\sum_{i \in \mathcal{I}(t)} n_i(t) = n(t)$. For each class $i \in \mathcal{I}(t)$, we use $k = 1, ..., n_i(t)$ to index the clients of this class at time t. Similar to the notations $C^*_{i(l),j}, \overline{C}_j^l(t)$ and $C_j^l(t)$, we denote $C^*_{ij}, \overline{C}_{ij}^k(t)$ and $C_{ij}^k(t)$ as the true expected value, empirical average of past payoffs at time t, and the UCB estimate at time t, respectively, for the payoff of server j serving tasks from the k-th client of the underlying class i. We also define $h_{ij}^k(t)$, $h_i^k(t) = \sum_{j=1}^J h_{ij}^k(t)$, and $p_{ij}^k(t)$ analogously to $h_j^l(t)$, $h^l(t)$ and $p_j^l(t)$. Thus, the UCB estimate (9) in Step 1 of our proposed Algorithm 1 is equivalent to

$$C_{ij}^k(t) \leftarrow \min\left\{\overline{C}_{ij}^k(t-1) + \sqrt{\frac{2\log h_i^k(t-1)}{h_{ij}^k(t-1)}},\ 1\right\}, \quad (13)$$

and the maximization problem (7) in Step 2 of our proposed Algorithm 1 is equivalent to:

$$\max_{[p_{ij}^k]\geq 0} \sum_{i\in\mathcal{I}(t)} \sum_{k=1}^{n_i(t)} \left\{ \frac{1}{V} \log\left(\sum_{j=1}^J p_{ij}^k\right) + \sum_{j=1}^J p_{ij}^k \left(C_{ij}^k(t) - \gamma\right) \right\} \quad (14)$$

$$\text{s.t.} \sum_{i\in\mathcal{I}(t)} \sum_{k=1}^{n_i(t)} p_{ij}^k \leq \mu_j, \quad \text{for all } j \in \mathcal{S}. \quad (15)$$

Our proof below will use these equivalent forms of the proposed Algorithm 1. In the rest of the analysis, we will use boldface variables (e.g., $\mathbf{n}$, $\mathbf{p}$, and $\mathbf{C}$) to denote vectors, and use regular-font variables (e.g., n_i, p_{ij}^k, and C_{ij}^k) to denote scalars. A list of notations is provided in Table I.

B. Handling Uncertain Client Dynamics

Recall that $\lambda_i = \lambda\rho_i$, $\lambda = \sum_{i=1}^I \lambda_i$ and $\mu = \sum_{j=1}^J \mu_j$. Define the vector $\mathbf{n}(t) = [n_i(t), 1 \leq i \leq I]$ and $\mathbf{p}(t) =$

TABLE I
LIST OF NOTATIONS

Symbol	Meaning
$C^k_{ij}(t)$	UCB estimate for server j by client k of class i
$C^l_j(t)$	UCB estimate for server j by client l
C^*_{ij}	True expected payoff for server j and class i
$\overline{C}^k_{ij}(t), \overline{C}^l_j(t)$	Empirical payoff average
$\gamma > 1$	Parameter in (7) and (14)
p^*_{ij}	Solution to upper bounds (4) or (17)
$p^k_{ij}(t)$, $p^l_j(t)$	Solution to (7) or (14) using UCB payoff estimates
$\widehat{p}^k_{ij}(t)$	Solution to (14) with $C^k_{ij}(t)$ replaced by C^*_{ij}

$[p^k_{ij}(t), i \in \mathcal{I}(t), 1 \le j \le J, 1 \le k \le n_i(t)]$. Define the Lyapunov function $L(\mathbf{n}(t))$ as

$$L(\mathbf{n}(t)) = \frac{1}{2}\sum_{i=1}^{I} \frac{n_i^2(t)}{\lambda_i}. \tag{16}$$

Recall that $[p^*_{ij}]$ is the optimal solution of the upper bound in (4). Note that if we replace each C^*_{ij} in (4) by $C^*_{ij} - \gamma$, this will result in the same optimal solution. Thus, $[p^*_{ij}]$ is also the optimal solution to the following optimization problem:

$$\max_{[p_{ij}]\ge 0} \sum_{i=1}^{I} \lambda_i \sum_{j=1}^{J} p_{ij}(C^*_{ij} - \gamma), \text{ subject to (5) and (6).} \tag{17}$$

Next, we will add a properly-scaled version of the following term to the drift of the Lyapunov function $L(\mathbf{n}(t+1)) - L(\mathbf{n}(t))$:

$$\Delta(t) = \sum_{i=1}^{I}\sum_{j=1}^{J} \lambda_i p^*_{ij}(C^*_{ij} - \gamma) - \sum_{i\in\mathcal{I}(t)}\sum_{k=1}^{n_i(t)}\sum_{j=1}^{J} p^k_{ij}(t)(C^*_{ij} - \gamma). \tag{18}$$

The value of $\Delta(t)$ captures the gap between the achieved payoff and the upper bound in (17), both adjusted by γ. The following lemma is the first step towards bounding the Lyapunov drift plus this payoff gap [30].

Lemma 3. *The expected drift plus payoff gap is bounded by*

$$\mathbb{E}\left[L(\mathbf{n}(t+1)) - L(\mathbf{n}(t)) + \frac{V}{N}\Delta(t) \mid \mathbf{n}(t), \mathbf{p}(t)\right] \le \frac{A_1(t)}{N} + \frac{V}{N}\sum_{i\notin\mathcal{I}(t)}\sum_{j=1}^{J} \lambda_i p^*_{ij}(C^*_{ij} - \gamma) + \frac{c_1 + c_2}{N}, \tag{19}$$

where $c_1 = \frac{I}{2}$, $c_2 = \frac{\mu^2}{2}\left(1 + \frac{\gamma^2}{(\gamma-1)^2 N}\right)\sum_{i=1}^{I}\frac{1}{\lambda_i}$, *and*

$$A_1(t) \triangleq \sum_{i\in\mathcal{I}(t)}\sum_{k=1}^{n_i(t)}\sum_{j=1}^{J}\left(\frac{n_i(t)}{\lambda_i} + V(C^*_{ij} - \gamma)\right)\left(\frac{\lambda_i p^*_{ij}}{n_i(t)} - p^k_{ij}(t)\right).$$

Proof of Theorem 2. Recall that in Step 2 of our proposed algorithm, we have chosen $\gamma > 1 \ge C^*_{ij}$. Hence, the second term on the right-hand-side of (19) is always less than 0. Then, taking expectations of (19) over $\mathbf{n}(t)$ and $\mathbf{p}(t)$, summing over $0 \le t \le T-1$, and divided by T, we have

$$\frac{1}{T}\mathbb{E}\left[L(\mathbf{n}(T)) - L(\mathbf{n}(0))\right] + \frac{V}{TN}\sum_{t=0}^{T-1}\mathbb{E}\left[\Delta(t)\right] \le \frac{1}{NT}\sum_{t=0}^{T-1}\mathbb{E}\left[A_1(t)\right] + \frac{c_1 + c_2}{N}.$$

Since the system at time $t = 0$ is empty, i.e., $n(t) = 0$, it follows that $L(\mathbf{n}(0)) = 0$. In view of $L(\mathbf{n}(T)) \ge 0$, the last displayed equation gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\Delta(t)\right] \le \frac{1}{TV}\sum_{t=0}^{T-1}\mathbb{E}\left[A_1(t)\right] + \frac{c_1 + c_2}{V}. \tag{20}$$

Let R_T denote the expected payoff per unit time achieved by Algorithm 1 for a given time T as defined in (3). By definitions of $\Delta(t)$, R_T, and R^*, we get that

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\Delta(t)\right] = R^* - R_T - \gamma\lambda + \frac{\gamma}{T}\sum_{t=0}^{T-1}\sum_{j=1}^{J}\mathbb{E}\left[\sum_{i\in I(t)}\sum_{k=1}^{n_i(t)} p^k_{ij}(t)\right]. \tag{21}$$

We show in [30, Appendix A.1] that the total number $D(t)$ of departures at time t satisfies

$$\mathbb{E}\left[D(t) \mid \mathbf{n}(t), \mathbf{p}(t)\right] \le \frac{1}{N}\sum_{i\in I(t)}\sum_{j=1}^{J}\sum_{k=1}^{n_i(t)} p^k_{ij}(t).$$

It follows that

$$\begin{aligned}\lambda - \frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=1}^{J}\mathbb{E}\left[\sum_{i\in I(t)}\sum_{k=1}^{n_i(t)} p^k_{ij}(t)\right] &\le \lambda - \frac{N}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[D(t)\right] \\ &= \frac{N}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[U(t+1) - D(t)\right] \\ &= \frac{N}{T}\mathbb{E}\left[n(T)\right] \le \frac{\beta_3 N(V+1)}{\gamma T},\end{aligned}$$

where $U(t)$ denotes the total number of arrivals at time t, and the last inequality holds in view of Theorem 1 and the definition of β_3 in (12). By combining the last displayed equation with (21) and (20), we get that

$$R^* - R_T \le \frac{\beta_3 N(V+1)}{T} + \frac{1}{TV}\sum_{t=1}^{T}\mathbb{E}\left[A_1(t)\right] + \frac{c_1 + c_2}{V}. \tag{22}$$

In the rest of this section, we show that for all T,

$$\frac{1}{TV}\sum_{t=1}^{T}\mathbb{E}\left[A_1(t)\right] \leq \frac{\lambda}{N}\big[4J\sqrt{2N\log N}+\mu\big(4\sqrt{2\log N}+3+\frac{3J\gamma^2}{(1-\gamma)^2}\big)\big]. \tag{23}$$

Substituting (23) into (22) and invoking the assumption $N\log N \geq 1$, the result of Theorem 2 readily follows.

□

The remainder of the section focuses on proving (23).

C. Bounding $A_1(t)$: Handling Payoff Uncertainty

The key in bounding $A_1(t)$ is to account for the impact of payoff estimation errors. In the rest of this subsection, we fix $\mathbf{n} = \mathbf{n}(t)$. Recall that $\mathcal{I}(t) = \{i|n_i(t) \geq 1\}$, and $\mathbf{p}(t) = [p_{ij}^k(t)]$ is the solution to the optimization problem (14). Denote vectors $\mathbf{W} = [W_{ij}^k, i \in \mathcal{I}(t), 1 \leq j \leq J, 1 \leq k \leq n_i(t)]$ and $\boldsymbol{\pi} = [\pi_{ij}^k, i \in \mathcal{I}(t), 1 \leq j \leq J, 1 \leq k \leq n_i(t)]$. We define function

$$f(\boldsymbol{\pi}|\mathbf{n},\mathbf{W}) \triangleq \sum_{i\in\mathcal{I}(t)}\sum_{k=1}^{n_i(t)}\left[\log\left(\sum_{j=1}^{J}\pi_{ij}^k\right)+V\sum_{j=1}^{J}(W_{ij}^k-\gamma)\pi_{ij}^k\right]. \tag{24}$$

When $\mathbf{W} = [C_{ij}^k(t)]$, the value of this function at $\boldsymbol{\pi} = \mathbf{p}(t)$ is precisely the optimal objective value of (14) multiplied by V. Now, let $\mathbf{C}^*$ denote the vector $[W_{ij}^k]$ such that $W_{ij}^k = C_{ij}^*$ for all $i \in \mathcal{I}(t), 1 \leq j \leq J, k = 1, ..., n_i(t)$. Denote $\widehat{\mathbf{p}}(t) = \left[\widehat{p}_{ij}^k(t)\right]$ as the maximizer of $f(\boldsymbol{\pi}|\mathbf{n},\mathbf{C}^*)$ over the constraint (15). We have the following lemma [30].

Lemma 4. *The following holds for each time-slot t:*

$$A_1(t) \leq f(\widehat{\mathbf{p}}(t)|\mathbf{n},\mathbf{C}^*) - f(\mathbf{p}(t)|\mathbf{n},\mathbf{C}^*) \tag{25}$$
$$\leq A_2(t) + A_3(t), \tag{26}$$

where

$$A_2(t) = V\sum_{i\in\mathcal{I}(t)}\sum_{k=1}^{n_i(t)}\sum_{j=1}^{J}\left(C_{ij}^k(t)-C_{ij}^*\right)p_{ij}^k(t),$$

$$A_3(t) = V\sum_{i\in\mathcal{I}(t)}\sum_{k=1}^{n_i(t)}\sum_{j=1}^{J}\left(C_{ij}^*-C_{ij}^k(t)\right)\widehat{p}_{ij}^k(t).$$

To appreciate the difference between the two terms in (25), recall that $\widehat{\mathbf{p}}(t)$ maximizes $f(\boldsymbol{\pi}|\mathbf{n},\mathbf{C}^*)$, while $\mathbf{p}(t)$ maximizes $f(\boldsymbol{\pi}|\mathbf{n},\mathbf{C}(t))$, where we have denoted $\mathbf{C}(t) = [C_{ij}^k(t)]$. Clearly, if there were no errors in the payoff estimates $\mathbf{C}(t)$, i.e., if $\mathbf{C}(t) = \mathbf{C}^*$, we would have obtained $\mathbf{p}(t) = \widehat{\mathbf{p}}(t)$ and $A_1(t) \leq 0$ trivially. This lemma thus bounds the difference even when $\mathbf{C}(t) \neq \mathbf{C}^*$.

It remains to upper bound $(1/T)\sum_{t=1}^{T}\mathbb{E}[A_2(t)]$ and $(1/T)\sum_{t=1}^{T}\mathbb{E}[A_3(t)]$ for a given time T. We define Λ as a particular realization of the sequence of client arrival-times up to time slot T. We use $\mathbb{E}_\Lambda$ to denote the conditional expectation given Λ. Given Λ, we slightly abuse notation and use the index k now to denote the k-th client of class i that arrives to the system. Let $t_1(k)$ denote the arrival time of this client, and $t_2(k)$ denote the *minimum* of her departure time and T.

Lemma 5. *Suppose that the k-th arriving client of the underlying class i brings a total number a_k of tasks into the system. For all $1 \leq j \leq J$, it holds that*

$$\mathbb{E}_\Lambda\left[\sum_{s=t_1(k)}^{t_2(k)}(C_{ij}^k(s)-C_{ij}^*)p_{ij}^k(s)\right] \leq 4\sqrt{2a_k\log a_k}+4\mu_j\sqrt{2\log a_k}+3\mu_j. \tag{27}$$

Lemma 6. *Suppose that the k-th arriving client of class i arrives at time $t_1(k)$. Then, the following holds,*

$$\mathbb{E}_\Lambda\left[\sum_{j=1}^{J}\sum_{s=t_1(k)}^{t_2(k)}\left(C_{ij}^*-C_{ij}^k(s)\right)\widehat{p}_{ij}^k(s)\right] \leq 3J\left(\frac{\gamma}{\gamma-1}\right)^2\mu. \tag{28}$$

By summing (27) and (28), respectively, over all users k, we can then obtain that

$$\frac{1}{TV}\sum_{t=1}^{T}\mathbb{E}\left[A_2(t)\right] \leq \frac{\lambda}{N}\left(4J\sqrt{2N\log N}+\left(4\sqrt{2\log N}+3\right)\mu\right). \tag{29}$$

and

$$\frac{1}{TV}\sum_{t=1}^{T}\mathbb{E}[A_3(t)] \leq \frac{3\lambda\mu J}{N}\left(\frac{\gamma}{1-\gamma}\right)^2. \tag{30}$$

Remark: The proofs of both lemmas involve finite-time regret analysis for UCB [8]. However, there is a major innovation. In standard MAB problems, exactly one arm is pulled at each time. In our system, the rate that servers are chosen is determined by $p_{ij}^k(t)$, which in turn depends on previous values of $C_{ij}^k(s)$, $s < t$. Our proofs use a delicate martingale argument to take care of such dependency. Between these two lemmas, the proof of Lemma 6 is even trickier. To see this, note that the loss in Lemma 5 is accumulated at the same rate as the rate that each server is chosen. Thus, we may view the time as being "slowed down" (compared to a standard MAB problem), and expect Lemma 5 to hold. In contrast, the loss in Lemma 6 is accumulated at the rate $\widehat{p}_{ij}^k(s)$, which is different from the rate $p_{ij}^k(t)$ that each server is chosen. Thus, a direct "slowing-down" argument will not work. Fortunately, by exploiting the structure of Algorithm 1, we can show that $\sum_{j=1}^{J}\widehat{p}_{ij}^k(t)$ and $\sum_{j=1}^{J}p_{ij}^k(t)$ cannot differ by more than a constant factor. We can then prove Lemma 6 using a similar martingale argument. For details, please refer to [30].

Finally, we remark that Lemma 5 is the main reason for the $\sqrt{\log N/N}$ regret in the second term of (11). In the classical MAB problem in [8], there is a non-zero gap δ between the best arm and the second-best arm. Thus, once the estimation error is within δ, which takes $\Theta(\log N)$ time slots, learning can stop. Hence, the regret is on the order of $\log N/N$. In contrast, in our problem the notions of "best/second-best arms" are more fluid because they depend on the number of clients in the system. Thus we do not have such a fixed gap δ, which is why we have a larger $\sqrt{\log N/N}$ loss due to learning.

VI. Conclusion

In this paper, we propose a new utility-guided task assignment algorithm for queueing systems with uncertain payoffs. Our algorithm seamlessly integrates online learning and adaptive control, without relying on any prior knowledge of the statistics of the client dynamics or the payoff vectors. We show that, compared to an oracle that knows all client dynamics and payoff vectors beforehand, the gap of the expected payoff per unit time of our proposed algorithm at any finite time T is on the order of $1/V + \sqrt{\log N/N} + N(V+1)/T$, where V is a parameter controlling the weight of a logarithmic utility function, and N is the average number of tasks per client. Our analysis carefully accounts for the closed-loop interactions between the queueing and learning processes. Numerical results indicate that the proposed algorithm outperforms a myopic matching policy and a standard queue-length based policy that does not explicitly address such closed-loop interactions.

There are a number of future research directions. First, a straightforward lower bound on the payoff loss is $O(\log N/N)$ [8]. Thus, it would be interesting to study whether the proposed algorithm can be further improved to reduce the second regret term of (11) to $O(\log N/N)$. Second, in order to limit the payoff-feedback delay, our algorithm assumes that the service of each server is deterministic. We plan to generalize to the case with random service or even unknown service rates. Finally, although our model does not assume any prior knowledge of the class-dependent payoff vectors, it would be interesting to study whether the idea from this paper can be used to improve the policies in [4], [5], [6] when such knowledge is available.

Acknowledgement

This work has been partially supported by the National Science Foundation through grants CCF-1755960, CNS-1457137, and ECCS-1509536.

References

[1] R. Combes, C. Jiang, and R. Srikant, "Bandits with budgets: Regret lower bounds and optimal algorithms," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 1, pp. 245–257, 2015.

[2] D. R. Karger, S. Oh, and D. Shah, "Budget-optimal crowdsourcing using low-rank matrix approximations," in *49th Annual Allerton Conference on Communication, Control, and Computing*, 2011, pp. 284–291.

[3] B. Tan and R. Srikant, "Online advertisement, optimization and stochastic networks," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2854–2868, 2012.

[4] R. Johari, V. Kamble, and Y. Kanoria, "Know your customer: Multi-armed bandits with capacity constraints," *arXiv preprint arXiv:1603.04549v1*, 2016.

[5] L. Massoulie and K. Xu, "On the capacity of information processing systems," in *Conference on Learning Theory*, 2016, pp. 1292–1297.

[6] R. Johari, V. Kamble, and Y. Kanoria, "Matching while learning," in *Proceedings of the 2017 ACM Conference on Economics and Computation (EC '17)*, Cambridge, MA, June 2017.

[7] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in Applied Mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

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

[9] J. Gittins, K. Glazebrook, and R. Weber, *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[10] P. Whittle, "Restless bandits: Activity allocation in a changing world," *Journal of Applied Probability*, vol. 25, no. A, pp. 287–298, 1988.

[11] D. Kalathil, N. Nayyar, and R. Jain, "Decentralized learning for multiplayer multiarmed bandits," *IEEE Transactions on Information Theory*, vol. 60, no. 4, pp. 2331–2345, 2014.

[12] A. Anandkumar, N. Michael, A. K. Tang, and A. Swami, "Distributed algorithms for learning and cognitive medium access with logarithmic regret," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 4, pp. 731–745, 2011.

[13] L. Lai, H. El Gamal, H. Jiang, and H. V. Poor, "Cognitive medium access: Exploration, exploitation, and competition," *IEEE Transactions on Mobile Computing*, vol. 10, no. 2, pp. 239–253, 2011.

[14] R. Combes and A. Proutiere, "Dynamic rate and channel selection in cognitive radio systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 5, pp. 910–921, 2015.

[15] S. Buccapatnam, J. Tan, and L. Zhang, "Information sharing in distributed stochastic bandits," in *IEEE INFOCOM*, 2015, pp. 2605–2613.

[16] T. L. Lai and Z. Ying, "Open bandit processes and optimal scheduling of queueing networks," *Advances in Applied Probability*, vol. 20, no. 2, pp. 447–472, 1988.

[17] A. Badanidiyuru, J. Langford, and A. Slivkins, "Resourceful contextual bandits," in *Conference on Learning Theory*, 2014, pp. 1109–1134.

[18] J. Feldman, N. Korula, V. S. Mirrokni, S. Muthukrishnan, and M. Pál, "Online ad assignment with free disposal," in *WINE*, vol. 9. Springer, 2009, pp. 374–385.

[19] L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks," *IEEE Trans. on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, December 1992.

[20] M. J. Neely, E. Modiano, and C. Li, "Fairness and Optimal Stochastic Control for Heterogeneous Networks," in *IEEE INFOCOM*, Miami, FL, March 2005.

[21] X. Lin, N. B. Shroff, and R. Srikant, "A Tutorial on Cross-Layer Optimization in Wireless Networks," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 24, no. 8, pp. 1452–1463, 2006.

[22] ——, "On the Connection-Level Stability of Congestion-Controlled Communication Networks," *IEEE Trans. on Inf. Theory*, vol. 54, no. 5, pp. 2317–2338, May 2008.

[23] G. De Veciana, T. J. Lee, and T. Konstantopoulos, "Stability and Performance Analysis of Networks Supporting Elastic Services," *IEEE/ACM Trans. on Networking*, vol. 9, no. 1, pp. 2–14, February 2001.

[24] T. Bonald and L. Massoulie, "Impact of Fairness on Internet Performance," in *Proceedings of ACM SIGMETRICS*, Cambridge, MA, June 2001, pp. 82–91.

[25] K. Bimpikis and M. G. Markakis, "Learning and hierarchies in service systems," *under review*, 2015.

[26] V. Shah, L. Gulikers, L. Massoulie, and M. Vojnovic, "Adaptive matching for expert systems with uncertain task types," *arXiv preprint arXiv:1703.00674*, 2017.

[27] M. Chiang, "To Layer or Not to Layer: Balancing Transport and Physical Layers in Wireless Multihop Networks," in *IEEE INFOCOM*, Hong Kong, March 2004.

[28] A. Eryilmaz and R. Srikant, "Fair Resource Allocation in Wireless Networks Using Queue-length-based Scheduling and Congestion Control," in *IEEE INFOCOM*, Miami, FL, March 2005.

[29] P. R. Kumar and S. P. Meyn, "Stability of Queueing Networks and Scheduling Policies," *IEEE Transactions on Automatic Control*, vol. 40, pp. 251–260, February 1995.

[30] W.-K. Hsu, J. Xu, X. Lin, and M. R. Bell, "Integrate Learning and Control in Queueing Systems with Uncertain Payoffs," Purdue University, Tech. Rep., 2017, available at https://engineering.purdue.edu/%7elinx/papers.html.