Meta-Scheduling for the Wireless Downlink Through Learning With Bandit Feedback

Jianhan Song, Gustavo de Veciana, *Fellow, IEEE*, and Sanjay Shakkottai, *Fellow, IEEE*

***Abstract*— In this paper, we study learning-assisted multi-user scheduling for the wireless downlink. There have been many scheduling algorithms developed that optimize for a plethora of performance metrics; however a systematic approach across diverse performance metrics and deployment scenarios is still lacking. We address this by developing a meta-scheduler – given a diverse collection of schedulers, we develop a learning-based overlay algorithm (meta-scheduler) that selects that "best" scheduler from amongst these for each deployment scenario. More formally, we develop a multi-armed bandit (MAB) framework for meta-scheduling that assigns and adapts a score for each scheduler to maximize reward (e.g., mean delay, timely throughput etc.). The meta-scheduler is based on a variant of the Upper Confidence Bound algorithm (UCB), but adapted to interrupt the queuing dynamics at the base-station so as to filter out schedulers that might render the system unstable. We show that the algorithm has a poly-logarithmic regret in the expected reward with respect to a genie that chooses the optimal scheduler for each scenario. Finally through simulation, we show that the meta-scheduler learns the choice of the scheduler to best adapt to the deployment scenario (e.g. load conditions, performance metrics).**

***Index Terms*— Online learning, bandit algorithms, upper confidence bound, wireless networks, scheduling.**

I. Introduction

MULTI-USER scheduling for wireless downlink systems is a particularly challenging task for two key reasons. First, mobile users and services may have diverse performance goals/requirements that should ideally be optimized over a wide variety of traffic loads/mixes and heterogeneous user service rates that can vary by over an order of magnitude. Second, because mobile users' see time-varying service rates, it is desirable to incorporate some form of opportunistic scheduling, favoring scheduling users when their service rates are high. To address these challenges wireless schedulers use a combination of the current channel conditions (e.g., obtained through channel quality feedback from mobile users) and current queue backlogs to dynamically assign users to channel resources so as to meet the desired various performance objectives including, e.g., throughput optimality (stability), mean packet/flow delay, delay tails, timely throughput, video quality of experience, etc.

Although a substantial number of scheduling algorithms have been proposed, solutions that are able to systematically address the above mentioned challenges are still lacking. Indeed, an algorithm best suited for a given scenario may depend on a variety of factors including traffic load/mix and users' channels, or more generally on the usage patterns associated with the time of day.[1] Moreover in some cases the desired performance metrics for a subset of users may not be easily pre-specified, e.g., measures of video quality, whence it is not clear what type of scheduler to deploy. Furthermore, even if one has access schedulers which are fine tuned to particular scenarios (e.g., learned through a reinforcement learning (RL) algorithm), we typically have no performance guarantees over the wide range of settings typical of wireless systems. Whence it is unclear that it is safe to deploy such scheduling policies.

In this paper, we propose a *meta-scheduler* – an online learning (bandit) algorithm which for a given operational scenario dynamically selects the best scheduler from a set of predefined policies (e.g., MaxWeight, Log rule, Exp rule, Priority rule, RL schedulers, etc.). The scheduler in turn, determines user-to-channel assignments. In our approach, scheduling policies are viewed as bandit arms, and the meta-scheduler dynamically chooses the scheduler (aka plays an arm) based on the mobile users' feedback. The goal is to provide a learning framework that efficiently identifies the best among a pre-selected set of state-of-the-art policies for a given underlying scenario (characterized by traffic, channel states, user metrics, etc.)

In adapting the bandit framework to our queueing setting, we need to address two challenges: *(i)* Arbitrarily switching among schedulers over time can lead to queue instability, even if each of the schedulers is stable. Indeed, one can show that switching between two MaxWeight schedulers with different weights can lead to unstable queues. *(ii)* If one or more of the possible schedulers is unstable for a given scenario (e.g. a round-robin scheduler in a high-load wireless setting), then a poor choice may lead to long term instability.

Our approach uses the fact that stable queueing systems typically exhibit cyclical sample-paths associated with busy periods for the overall system. Under appropriate assumptions, the queue dynamics in a busy period are conditionally (given the scheduling policy) independent. Our meta-scheduler thus determines which scheduler (arm to play) only at the beginning

Manuscript received June 24, 2020; revised December 12, 2020 and May 5, 2021; accepted September 5, 2021; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor L. Huang. Date of publication October 14, 2021; date of current version April 18, 2022. This work was supported in part by NSF under Grant CNS-1731658, Grant CNS-1718089, and Grant CNS-1910112; in part by the Army Futures Command Grant W911NF1920333; and in part by the Wireless Networking and Communications Group Industrial Affiliates Program. An earlier version of this paper appeared in the Workshop on Machine Learning and Communications (WMLC) at WiOpt 2020 [1]. *(Corresponding author: Jianhan Song.)*

The authors are with the Department of Electrical and Computer Engineering, Cockrell School of Engineering, The University of Texas at Austin, Austin, TX 78712 USA (e-mail: jianhansong@utexas.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2021.3117783, provided by the authors.

Digital Object Identifier 10.1109/TNET.2021.3117783

[1] As an example, consider a set of users with different latency requirements. When the traffic load is low, it may be desirable to give scheduling priority to users with stricter packet deadlines without degrading other users' performance; when the load increases, however, a fairer scheduler may be preferred.

of cycles and the chosen scheduler is maintained for the duration of the cycle ensuring independent reward samples across cycles). Further to ensure that cycles do not have infinite durations, the meta-scheduler interrupts[2] cycles that have exceedingly long durations. These decisions have to be properly designed such that cycles due to unstable schedulers (which have unbounded cycle lengths) are played infrequently, and when played, get interrupted (truncated) as soon as possible. Further "good" cycles associated with stable schedulers should not get interrupted. As we will see, designing a sound interruption mechanism in conjunction with online learning through bandit feedback is crucial designing a meta-scheduler which achieves a low regret with respect to a genie algorithm (baseline that always plays the best/highest-reward scheduler for a particular scenario).

Finally, it is worth noting that there are possibly two parallel methodologies to systemically address the scheduling problem for various environments/applications. The first one is to formulate scheduling as an MDP problem for any given performance metric, and utilize a data-driven method to train and learn the optimal strategy for the given model, i.e., the reinforcement learning approach. Although this is a powerful method to generate *new* schedulers, the training process typically requires considerable exploration of different regimes and a large computational effort, and is often conducted offline (i.e., before deployment). Therefore, it is essential to properly model the performance metric and traffic environment where the scheduler will be deployed, which further raises difficulty and safety concerns (due to the mismatch between the training and deployment environments). By contrast, in this paper we tackle the problem from the second perspective. We will answer the following question: Given a set of *existing* policies (which could include one or more pre-trained RL schedulers optimized for specific settings), how one can determine the best scheduler *among these candidates* without assuming prior knowledge on the current environment/performance metric. Our bandit framework learns in an online manner with light-weight computation and relatively low convergence time needed, and can in principle keep re-optimizing to changing scenarios (through re-running the meta-scheduling algorithm when the environment significantly changes). When the optimal policy is unclear, which is common in many real applications given all the uncertainties, our approach transfers the burden of choosing the best suited *environment-specific state-of-the-art* scheduler to a learning algorithm.

A. Contribution

Our main contributions are the following:

- *Meta-scheduler:* We develop a meta-scheduler algorithm based on (UCB + Interruptions). At the beginning of each queuing cycle, the meta-scheduler determines a scheduler to be used for that cycle using a variant of the Upper Confidence Bound (UCB) Algorithm. This consists of (i) determining a score for each scheduler (empirical reward + confidence bonus) that is multiplied by an indicator that estimates if each scheduler is stable (meaning the cycle times are finite), and choosing the scheduler with the highest score; and (ii) determining an interrupt threshold for the cycle, at which time all packets in the queues are dropped if the cycle has not ended before then.
- *Theoretical guarantees:* For the meta-scheduler, we show that the regret (expected cumulative difference in reward) with respect to a genie algorithm that chooses the optimal (highest expected reward) scheduler scales as $O(\log n)$, where n is the number of cycles[3] and correspondingly $O(\log^2 \tau)$ where τ is the time-slot index. Further, the expected number of packets dropped due to interruptions also scales as $O(\log^2 \tau)$. When packet drop is forbidden, an alternative mechanism to clear up the queueing system is introduced at a slight expense of the total regret.
- *Simulation Results:* We simulate the meta-scheduler in a variety of wireless settings. These include different reward for performance metrics such as mean delay, delivering packets on time, and penalizing bursty service, and various schedulers including the MaxWeight, Exp, Log, max-rate and round-robin and opportunistic priority, and different load conditions. Our simulations show that as conditions vary (e.g. different loads, or different performance metrics), the meta-scheduler adapts to choose a different scheduler that maximizes the reward for each scenario.

B. Related Work

Wireless Scheduling. The design of multi-user wireless schedulers has received substantial attention, see e.g., [2] and references therein. For infinitely backlogged user queues researchers have devised various classes of opportunistic schedulers that optimize the sum user utility (fairness criteria) of their long-term throughputs or so called timely throughput, see e.g., [3]–[7]. For settings where user queues are subject to stochastic arrivals e.g., packet streams, initial work focused on characterizing *throughput-optimal* schedulers which ensure queue stability if indeed stability can be achieved without prior knowledge of the traffic load and service capacity. These include, for example the MaxWeight rule [8], [9], Exp rule [10] and Log rule [11], which in addition to throughput optimality achieve different user-level performance objectives. Meanwhile, non-throughput-optimal policies can in certain load scenarios provide better performance, e.g., max-rate, proportionally fair, round-robin and the priority-based rules. Although there is substantial work in this area, the question of how to realize the best performance tradeoffs among heterogeneous users with diverse performance goals remains open and challenging.

Not surprisingly recently, *reinforcement learning* (RL) approaches have been proposed to address complex scheduling problems, including job scheduling for data centers [12] and wireless scheduling in various settings [13]–[16]. RL algorithms provide a general approach to determine good schedulers for specific scenarios and possibly, but substantially more challenging, ones that are good for a range scenarios in terms of the user traffic, service capacity and or performance objectives. Despite showing great potential in several applications, providing theoretical performance guarantees for RL based schedulers remains an open question. Limited success has been

[2]A cycle is interrupted by forcibly making all queues to be zero, e.g., by dropping packets in the buffers.

[3]The regret scaling is slightly weaker under weaker assumptions on the cycle tail distributions, please see Section IV for details.

achieved in some simple settings (in terms of traffic model or user metrics) – see e.g., [17]–[19]. However, advanced RL methods, especially those involving neural networks which have attracted the most attention in practice, typically lack rigorous performance guarantees, and thus it is unclear they are safe to deploy. The goal of this paper differs from the common focus of designing practically or theoretically good RL schedulers in current RL-networking literature. Instead, our framework aims at better utilizing the knowledge of existing schedulers (including RL schedulers) to address various scheduling scenarios (in particular those with complicated performance metrics or traffic models), while ensuring queueing stability.

In the literature, some authors proposed scheduling policies that utilize online *statistical learning*, e.g., [20], [21], which involve learning system statistics online to improve performance of certain schedulers. We note that the above methodology is different from the bandit-based online learning framework we propose in this paper. Our framework is adaptive to statistics, but by learning the best scheduler among a predefined candidate set of policies for a specific scenario, instead of refining specific policies.

Multi-armed Bandits. Multi-armed Bandits (MAB) problems have been studied for many decades, with applications to clinical trials, recommendation systems and online advertising; see [22] and [23] for a comprehensive discussion on the state-of-art. In our model, each time we choose a new arm, the corresponding (random) cycle time can be interpreted as a cost. Such problems where each action costs non-unit amount of resources is referred to as *budgeted bandits*. Unlike classical MAB settings, the regret is not parameterized by a time horizon; instead the regret parameterization (and thus, the best arm) involves both the reward and cost variables, which significantly increases the complexity of the problem. This line of work was started by [24] and has been followed in many directions by [25]–[27].

A recent study on budgeted bandits in [28] introduces the idea of MAB with interruptions. At each time, a server works on a single task that has a heavy-tailed service completion time. A task can be interrupted if it is taking too long (but with loss in reward). The authors in [28] develop a variant of the Upper Confidence Bound (UCB) algorithm [29] that selects over (a finite set of) tasks as well as a finite set of task interrupt thresholds to discard ongoing tasks, i.e. arms are (task, interrupt-threshold) pairs. Their motivation is to interrupt a task that takes too long so as to start a new one to collect more rewards, and thereby benefit the total reward. Our model is inspired by their work, but significantly differs in the way that we deal with interruptions. In contrast to [28], our goal is to eventually avoid any interruptions, thus, we do not treat interruptions as arms of a bandit. Instead, we dynamically increase the threshold for each task (aka scheduling policy) to ensure we quickly filter out unstable policies for which the cycle times are infinite, while leaving stable policies (eventually) uninterrupted. Algorithmically, our approach modifies UCB with a *multiplicative censoring* that penalizes interruptions from occurring too often, which ensures that unstable arms (with infinite expected cycle completion times) are aggressively eliminated.

Finally, bandit algorithms have also been applied to wireless resource allocation problems more broadly. These include studies in cognitive radio probing [30], spectrum access [31], decentralized wireless computing [32], [33] and most recently, cellular scheduling [34]. An earlier version of this paper was presented in [1].

C. Notation

Throughout this paper, we use characters in bold font to denote vectors and normal font to denote scalars. Random variables are indicated by capital letters unless stated otherwise. We adopt the following technical abbreviations: "*w.h.p.*" for "with high probability", "*a.s.*" for "almost surely" and "*i.i.d.*" for "independently and identically distributed". Finally, we use $\mathbb{1}$ for the $\{0,1\}$ indicator function.

II. Model Settings

In this section, we consider a multi-arm bandit model for the wireless scheduling problem. The goal is to formulate a *meta-scheduler* that can explore different scheduling policies and learn in an online manner which among the candidate policies is the best, given a certain performance metric. Before introducing the meta-scheduler in detail, we first describe the traffic model and then describe the system from a perspective of regenerative processes. We will see it is natural to allow the meta-scheduler to switch policies only when the system "regenerates". Formal definitions of a *meta-policy* (policy of a meta-scheduler) and its regret are given at the end of this section.

A. Traffic and Service Model

We consider a packet-based queuing system with a set of u different users, denoted by $\mathcal{U}$, and a single server (base station). The system operates in discrete time slots. For simplicity, suppose all packets have the same size. At any time t, define the random vector $\boldsymbol{Q}[t] = (Q_1[t], \cdots, Q_u[t]) \in \mathbb{Z}_+^u$, where $Q_i[t]$ denotes the number of packets of the i-th user at the beginning of time slot t.

The random packet arrivals at time t are denoted by $\boldsymbol{A}[t] = (A_1[t], \cdots, A_u[t])$ where $A_i[t]$ has a integer-valued distribution bounded by $\bar{a}$ for any user $i \in \mathcal{U}$. We assume $(\boldsymbol{A}[t])_{t\geq 0}$ are *i.i.d.* across time and denote its expectation by $\boldsymbol{\lambda}$. The wireless channels' service rates at time t are modeled by a random vector $\boldsymbol{S}[t] = (S_1[t], \cdots, S_u[t])$ where $S_i[t]$ denotes the service rate available to the i-th user at t. $(\boldsymbol{S}[t])_{t\geq 0}$ are *i.i.d.* over time and also independent of the queue lengths and arrival process. A scheduling *policy* will decide which user to serve at each time slot based on the queue and channel state.

Let $\mathcal{C}$ denote the long-term capacity of the system (see [2]). This means for any arrival rate that lies in $\mathcal{C}^o$ (the interior of $\mathcal{C}$), there exists at least one policy that stabilizes the system (the average queue lengths are finite). We require $\boldsymbol{\lambda} \in \mathcal{C}^o$. We say a policy is *stable* (with respect to $\boldsymbol{\lambda}$) if it stabilizes the system.

Now suppose there is a finite set of scheduling policies (or *arms* in the bandit context), denoted by $\mathcal{A}$. For a fixed $\boldsymbol{\lambda} \in \mathcal{C}^o$, $\mathcal{A}$ consists of both stable and unstable policies, denoted by $\mathcal{A}^s(\boldsymbol{\lambda})$ and $\mathcal{A}^u(\boldsymbol{\lambda})$. Assume that $\mathcal{A}^s(\boldsymbol{\lambda}) \neq \phi$.

B. Regenerative Dynamics

Suppose the arrival always occurs right after the beginning of a slot while the transmission happen right before the end

of a slot. We say the system *returns idle* when the sum of users' queue lengths is down to 0 from some positive value at the end of a time slot. A *cycle* is defined as the interval of time slots between two consecutive points in time when the system returns idle.[4] Further, without loss of generality, we assume the system starts empty at the beginning of the first slot. We can describe the system's dynamics based on such cycles as follows. The notation and definitions in this section follows [28], with appropriate modifications to reflect our setting.

Each arm k is associated with a stochastic process $((C^{(k)}(n), \boldsymbol{U}^{(k)}(n)))_{n\geq 1}$ where n denotes the index of cycles. If arm k is implemented after n-th time the system returns idle, the system observes a random cycle length $C^{(k)}(n)$ (before it returns idle again), and receives a sequence of *non-negative* rewards $\boldsymbol{U}^{(k)}(n) = (U^{(k)}(n,i) : i = 1, 2, \cdots, C^{(k)}(n))$ for each time slot in the cycle. Note that $C^{(k)}(n)$ for $n \geq 1$ are *i.i.d.* and $C^{(k)}(n) \geq 1$ $a.s.$.

Remark 1: To be precise, in order to make $C^{(k)}(n)$ i.i.d. over cycles, in addition to i.i.d. arrivals and channels assumed in our traffic and service model, it is necessary to assume the scheduling policy k only uses information associated with its current cycles. For example, a Markovian policy which chooses a service vector at time t only based on the current system states (e.g., $\boldsymbol{S}[t]$ and $\boldsymbol{Q}[t]$), such as MaxWeight, Log rule, etc., naturally satisfies this requirement. For a policy that keeps internal states which utilize information from the past, e.g., a proportionally fair scheduler using an exponential moving average of past throughput, we need to additionally reset internal states when a cycle begins.

We consider a reward scheme where the generated rewards are *i.i.d.* over cycles and grow no faster than linearly with corresponding time, which is formally stated in the next assumption.

Assumption 1: The cycle reward sequence $(\boldsymbol{U}^{(k)}(n))_{n\geq 1}$ for all $k \in \mathcal{A}$ are i.i.d. over n, and such that for $l = 1, \ldots, C^{(k)}(n)$,

$$0 \leq \sum_{i=1}^{l} U^{(k)}(n,i) \leq \bar{r}l, \quad \textit{almost surely} \tag{1}$$

for some $\bar{r} > 0$.

Remark 2: This assumption holds, for instance, if each packet is associated a bounded reward (e.g., over $[0,1]$) upon reception, and the cumulative reward over a time period is thus bounded by the maximal number of packets transmitted within that period, i.e., $\bar{r} = \bar{a}u$. In practice, for example, the cycle reward evaluated by a latency-sensitive user can be the number of packets that arrive on time within a cycle. It should be noted that the manner in which rewards are calculated/defined is not necessarily known by the base station in our model, which allows for more flexible user-customized reward schemes.

We denote the (total) cycle reward by $U^{(k)}(n) = \sum_{i=1}^{C^{(k)}(n)} U^{(k)}(n,i)$. Thus, it follows that $U^{(k)}(n)$ for $n \geq 1$ are *i.i.d.* across cycles and bounded as follows:

$$0 \leq U^{(k)}(n) \leq \bar{r}C^{(k)}(n), \quad \textit{almost surely}. \tag{2}$$

[4]In technical terms, a cycle consists of an *idle period* plus a *busy period*. When the system stays empty for a whole time slot, this slot is part of the idle period rather than a new cycle.

One question regarding the process is how frequently a policy forces the system to finish a cycle, i.e., the distribution of $C^{(k)}(n)$, which is vital for the meta-scheduler discussed in the sequel. When k is a stable arm, we have $\mathbb{P}(C^{(k)}(n) < \infty) = 1$ and the system will start a new cycle infinitely often. In addition, we have the following assumption on the cycle length of a stable arm.

Assumption 2: For a given $\boldsymbol{\lambda} \in \mathcal{C}^o$, we assume if arm $k \in \mathcal{A}^s(\boldsymbol{\lambda})$, $C^{(k)}(n)$ is a sub-exponential random variable. This implies that, there exist (possibly $\boldsymbol{\lambda}$-dependent) non-negative parameters (ν_k^2, α_k), such that for all $n \geq 1$,

$$\mathbb{P}(|C^{(k)}(n) - \mathbb{E}[C^{(k)}(n)]| \geq \varepsilon) \leq \begin{cases} 2e^{-\varepsilon^2/(2\nu_k^2)} & 0 < \varepsilon \leq \dfrac{\nu_k^2}{\alpha_k}, \\ 2e^{-\varepsilon/(2\alpha_k)} & \varepsilon > \dfrac{\nu_k^2}{\alpha_k}. \end{cases} \tag{3}$$

Remark 3: This assumption implies that for stable arms $k \in \mathcal{A}^s(\boldsymbol{\lambda})$, $C^{(k)}(n)$ has a light tail on the right (the left side is bounded). When the system has bounded arrival and channel distributions, and the policies considered are Markovian, this assumption holds true following an argument of [35]. One can then show that the empirical average $(1/n)\sum_{i=1}^{n} C^{(k)}(i)$ is sub-exponential with parameters $(\nu_k^2/n, \alpha_k/n)$. By Assumption 1, i.e., (2), $U^{(k)}(n)$ is also sub-exponential (with possibly larger parameters). Without loss of generality, we will assume both $C^{(k)}(n)$ and $U^{(k)}(n)$ are (ν_k^2, α_k)-sub-exponential, assuming the rewards are properly normalized.

If an unstable arm is applied, however, the system is transient and there is a chance that the system will never start a new cycle as $\mathbb{P}(C^{(k)}(n) = \infty) > 0$ for all $k \in \mathcal{A}^u(\boldsymbol{\lambda})$. This suggests that an additional stopping mechanism is needed when an unstable arm is explored by the meta-scheduler.

When $k \in \mathcal{A}^s(\boldsymbol{\lambda})$, observe that $\big((C^{(k)}(n), U^{(k)}(n))\big)_{n\geq 1}$ form a well-defined renewal-reward process. We next define the *renewal reward rate* of a stable policy.

$$r^{(k)} = \frac{\mathbb{E}[U^{(k)}(1)]}{\mathbb{E}[C^{(k)}(1)]} \quad \forall k \in \mathcal{A}^s(\boldsymbol{\lambda}). \tag{4}$$

By Renewal Theory, this rate captures the rate of rewards generated by a policy.

C. Meta-Scheduler, Feedback, and Interruptions

A meta-scheduler makes decisions on which arms to use and when, so as to maximize the rate of rewards of the system. In this paper, we will only consider meta-schedulers that comply with the following rules:

(1) A meta-scheduler can switch to another arm when the system returns idle;

(2) A meta-scheduler can *interrupt* a cycle, i.e., discarding all packets currently in the system and forcing the system to start a new cycle, so as to prevent unstable arms from occupying the system indefinitely. Furthermore, as in [28], we only consider conditions triggering such interruptions solely based on cycle time: a cycle gets interrupted when its length exceeds a threshold pre-selected before the cycle starts.

Remark 4: To allow for simpler analysis, we require that all packets to be discarded when a cycle is interrupted. As is shown later, a good meta-scheduler should be designed such that this event occurs rarely. If such packet drops are

unacceptable, instead of interrupting and dropping, we can switch to a default policy that is guaranteed to be stable (e.g., MaxWeight) when an interruption is triggered, and a cycle can be restarted when the system returns to the idle state. Later we will show that the loss of rewards induced by this extra process grows logarithmically in time.

There are several advantages in adopting those two rules. First, even scheduling policies that might result in unstable queues can be added to the mix, since the interruptions ensure that cycle times remain bounded. Moreover, they simplify the design of a meta-scheduler, since the system can be fully characterized by cycle lengths and rewards, i.e., the collection of processes $\{((C^{(k)}(n), \boldsymbol{U}^{(k)}(n)))_{n\geq 1} : k \in \mathcal{A}\}$, from the meta-scheduler's point of view regardless of how the actual queues and channels vary with time. This guarantees the independence of statistics for different arms and allows us to apply classical MAB methodologies. Furthermore, such a meta-scheduler preserves properties of regenerative processes that help analysis.

According to the rules mentioned above, a meta-scheduler can only make a *decision* when the system returns idle, which consists of two selections: the arm and the interruption threshold. Formally, we let $\pi = (\pi_n)_{n\geq 1}$ be a *meta-policy* (policy of a meta-scheduler), where $\pi_n = (A_n, L_n) \in \mathcal{A}\times(\mathbb{Z}^+\cup\{+\infty\})$. A decision $\pi_n = (k, l)$ implies that arm k is selected for n-th cycle, and the cycle will be interrupted immediately if it lasts over l time slots.

In order to model cycles under our interruption policy, we let $\hat{C}^{(k,l)}(n) = \min[C^{(k)}(n), l]$ and $\hat{\boldsymbol{U}}^{(k,l)}(n) = (U^{(k)}(n,i) : i = 1, 2, \cdots, \hat{C}^{(k,l)}(n))$. The *observed* (total) cycle reward $\hat{U}^{(k,l)}(n) = \sum_{i=1}^{\hat{C}^{(k,l)}(n)} U^{(k)}(n,i)$. Note that it still holds that $0 \leq \hat{U}^{(k,l)}(n) \leq \bar{r}\hat{C}^{(k,l)}(n)$ almost surely.

If $\pi_n = (k, l)$, we assume stochastic feedback Z_n is received for n-th cycle by the meta-scheduler as follows,

$$Z_n = (\hat{C}^{(k,l)}(n), \hat{U}^{(k,l)}(n), \mathbb{1}\{\hat{C}^{(k,l)}(n) < C^{(k)}(n)\}).$$

An illustration of the meta-policy dynamics is shown in Figure 1. Note that the reward for each single time slot is not required in the feedback. This suggests that if performance is evaluated at the user side, additional communication cost only occurs at the end of a cycle.

We assume π_n is solely based on the history of actions and feedback up to the decision. Thus, an *admissible meta-policy* considered in this paper is formally defined as follows. This is analogous to a similar notion in [28].

Definition 1 (Admissible Meta-Policy): We call a meta-policy $\pi = (\pi_n)_{n\geq 1}$ admissible if $\pi_n \in \mathcal{F}_n$ where $\mathcal{F}_n := \sigma(\pi_1, Z_1, \pi_2, Z_2, \cdots, \pi_{n-1}, Z_{n-1})$ is the σ-field induced by all the random decisions and feedback before n-th cycle.

Our goal is to design a good meta-policy that satisfies the following two objectives: (1) it suffers *negligible throughput loss*, i.e., the number of packets discarded due to interruptions by the meta-scheduler is sub-linear in time, and (2) it has a *sub-linear regret* over a given time horizon. We will define the regret in the next section.

D. Regret

As in the traditional MAB setting, we are interested in the *regret* of a meta-policy as compared to an optimal over a given time horizon τ. The regret for the meta-policy π stems from two reasons: (i) playing suboptimal arms (schedulers), and (ii) interrupting ongoing cycles. To formally define the regret, we follow a similar approach as in [28]. First, note that the number of cycles within a time horizon τ is a random variable, which can be viewed as a counting process.

Definition 2 (Counting Process): Consider a meta-policy π that is admissible. The total time of the first n-th cycle can be written as

$$S_n^\pi = \sum_{i=1}^{n} \sum_{(k,l)\in\mathcal{A}\times\mathbb{Z}^+} \mathbb{1}\{\pi_s = (k,l)\}\hat{C}^{(k,l)}(i).$$

Define a counting process $(N_\pi[\tau])_{\tau\geq 1}$ as follows.

$$N_\pi[\tau] = \max\{n : S_n^\pi \leq \tau\}.$$

Note that $N_\pi[\tau]$ indicates the number of completed cycles within time horizon τ.

Definition 3 (Cumulative Reward): Given a time horizon τ, the cumulative reward for an admissible meta-policy π is a random variable given as follows. (Denote $\tilde{N} := N_\pi[\tau]$ for notation simplicity.)

$$\mathrm{Rew}_\pi[\tau] = \sum_{i=1}^{\tilde{N}} \sum_{(k,l)} \mathbb{1}\{\pi_i = (k,l)\}\hat{U}^{(k,l)}(i) + \sum_{(k,l)} \mathbb{1}\{\pi_{\tilde{N}+1} = (k,l)\} \sum_{j=1}^{\tau - S_{\tilde{N}}^\pi} U^{(k)}(\tilde{N}+1, j). \quad (5)$$

The cumulative reward is the sum of (observed) cycle rewards from the first $N_\pi[\tau]$ completed cycles and the reward from the next uncompleted cycle up to time τ.

We call a meta-policy *simple-static* if the meta-scheduler consistently selects an arm with no cycle interruption. Let $\pi^{(k)}$ be the simple-static meta-policy selecting arm k, i.e., $\pi_n^{(k)} = (k, +\infty), \forall n \geq 1$. In this paper, we define the regret with respect to the best simple-static meta-policy π^{opt} that is stable and generates the most rewards (in expectation) within a given time. By the renewal theorem, $\lim_{\tau\to\infty} \mathrm{Rew}_{\pi^{(k)}}[\tau]/\tau = r^{(k)}$ $a.s.$ for all $k \in \mathcal{A}^s(\boldsymbol{\lambda})$. This implies that $\pi^{\mathrm{opt}} = \pi^{(k^*)}$ where $k^* = \mathrm{argmax}_{k\in\mathcal{A}^s(\boldsymbol{\lambda})} r^{(k)}$. The regret is formally defined as follows.

Definition 4 (Cumulative Regret): Let π^{opt} be the optimal simple-static meta-policy, i.e.

$$\pi_n^{\mathrm{opt}} = (k^*, \infty), \quad \forall n \geq 1 \quad (6)$$

where $k^* = \mathrm{argmax}_{k\in\mathcal{A}^s(\boldsymbol{\lambda})} r^{(k)}$. The regret of meta-policy π with respect to π^{opt} over any time horizon τ is defined as

$$\mathrm{Reg}_\pi[\tau] = \mathbb{E}[\mathrm{Rew}_{\pi^{\mathrm{opt}}}[\tau] - \mathrm{Rew}_\pi[\tau]]. \quad (7)$$

In the remaining sections, we will simply refer to k^* as the *optimal arm* (assumed to be unique). For notation simplicity, we suppress k^* as a single asterisk in the superscript when there is no ambiguity (e.g., $r^{(*)} := r^{(k^*)}$).

III. UCB Meta-Scheduler With Interruption

To guarantee negligible throughput loss and a sub-linear regret as discussed in Section II, the meta-scheduler should wisely select the arms and interruption thresholds such that the optimal arm is being applied at most of the time, and the packet discard hardly occurs. This implies the following guidelines when designing the algorithm: 1) the number of times a suboptimal arm (either unstable or stable) gets selected should be sub-linear in time; and 2) the unstable arms'

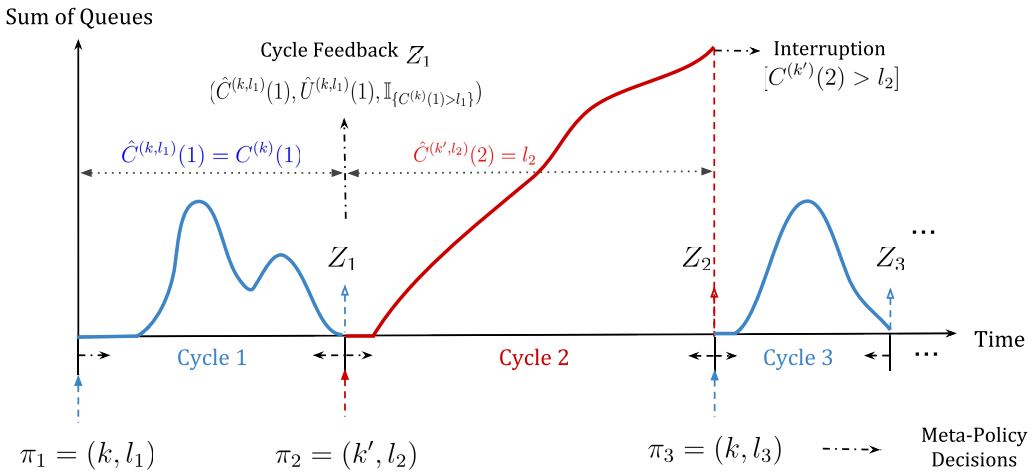


Fig. 1. Illustration of a meta-policy selecting arms from $\mathcal{A} = \{k, k'\}$. At the start of the process, the meta-scheduler makes decision $\pi_1 = (k, l_1)$, and receives feedback Z_1 after the system experiences a full cycle. Then the meta-scheduler decides $\pi_2 = (k', l_2)$, but has to interrupt the cycle as the system does not return idle before the cycle time reaches l_2. The meta-scheduler then collects feedback Z_2 and starts a new cycle with $\pi_3 = (k, l_3)$.

(possibly infinitely) long cycles must be stopped, while the cycles of the optimal arm should be preserved with little interruption. Motivated by these guidelines, we propose a UCB-type meta-scheduler with a properly-designed interruption rule.

To simplify notation and avoid ambiguity, let $C_s^{(k)}$ and $\hat{C}_s^{(k,l)}$ ($U_s^{(k)}$ and $\hat{U}_s^{(k,l)}$) be the full and observed cycle length (reward) of arm k *when it is selected the s-th time* (we call it s-th *sample* of k). Denote by $T_n^{(k)}$ as the number of times arm k has been chosen in the first n decisions. Thus, if $A_n = k$,

$$(C^{(k)}_{T_n^{(k)}}, U^{(k)}_{T_n^{(k)}}) = (C^{(k)}(n), U^{(k)}(n)).$$

Similar to a classical UCB algorithm, the meta-scheduler learns the arm statistics by keeping track of the empirical averages of cycle lengths and rewards. We formally define the *empirical rate* of arm k after s samples as $\hat{R}_s^{(k)}$. For all $s \geq 1$,

$$\hat{R}_s^{(k)} = \frac{\sum_{i=1}^{s} \hat{U}_i^{(k,F_i^{(k)})}}{\sum_{i=1}^{s} \hat{C}_i^{(k,F_i^{(k)})}}, \tag{8}$$

where $F_i^{(k)}$ denotes the threshold level for arm k's i-th sample. As a convention, the empirical rate equals 0 when $s = 0$. Let $\hat{R}^{(k)}(n) := \hat{R}^{(k)}_{T_n^{(k)}}$ be the empirical rates for the k-th arm *after* n-th cycle in the system.

As an overview, we present a simplified version of our meta-scheduler in Algorithm 1. The mathematical design of key variables will be discussed in a rigorous manner in the next section. Before that, let us first give some intuition as follows.

First, we observe that to avoid constantly interrupting a stable arm, it is necessary (and sufficient) to apply an interruption rule where the threshold of each arm is set to slowly grow with the number of samples (note that otherwise a fixed threshold will always result in linear throughput loss). Hence, we define a threshold function f_s as in line 2. For *any* arm k, we will use f_s as the interruption threshold for its s-th sample. With this design, the expected number of interruptions imposed on the optimal arm can be bounded by a *constant* if β and κ are large enough.

Algorithm 1 UCB Based Meta-Scheduler With Interruption

1: **Input:** Set of scheduling policies $\mathcal{A}$.
2: **Threshold Function**: $f_s := \beta + \kappa \log s$.
▷ β and κ to be defined
3: **Initialization**: Run every arm $k \in \mathcal{A}$ once with interruption threshold β, then initialize empirical rate $\hat{R}_1^{(k)}$.
4: **for** $n = |\mathcal{A}| + 1, |\mathcal{A}| + 2, \cdots$ **do**
5: [*before cycle* n]
6: **for all** $k \in \mathcal{A}$ **do**
7: Compute *Exploration Bonus* $\Delta_n^{(k)}$.
8: Compute *Stability Indicator* $I_n^{(k)}$.
▷ $I_n^{(k)}$: a Boolean variable
9: **Arm decision:**
$$A_n \in \operatorname*{argmax}_{k \in \mathcal{A}} I_n^{(k)} \times (\hat{R}^{(k)}(n-1) + \Delta_n^{(k)}).$$
10: **Cycle interruption decision:**
$$L_n = f_{T_n^{(A_n)}}.$$
11: [*after cycle* n]
12: Observe cycle feedback $\hat{U}^{(A_n,L_n)}(n), \hat{C}^{(A_n,L_n)}(n)$, then update $\hat{R}^{(A_n)}(n)$.

The meta-scheduler starts with running each policy once with an initial interruption level β and initializing the empirical rate $\hat{R}_1^{(k)}$ for any $k \in \mathcal{A}$. After this initialization phase, before each decision, the meta-scheduler will compare the "score" (i.e., *upper confidence bound*) for each of its arms. The score is the sum of its empirical reward rate and an *exploration bonus* (line 7). The exploration bonus is used to compensate for possibly under-performing empirical rate estimates in order to ensure adequate exploration before finding committing to optimal arm. We will show that *w.h.p.*, $\hat{R}^{(*)}(n-1) + \Delta_n^{(*)} > r^{(*)}$ for any $n \geq |\mathcal{A}|+1$. Meanwhile, the score of a *suboptimal stable* arm will be below $r^{(*)}$ after it is sufficiently explored.

Moreover, we design a *stability indicator* (line 8) to eliminate unstable arms by utilizing accumulated interruptions as a signal indicating whether an arm frequently induces long cycles. We keep track of the number of times each arm is interrupted, and $I_n^{(k)}$ is set 0 only if the total number of interruptions exceeds a limit (which is a function of n).

After computing the exploration bonus and the stability indicator, the meta-scheduler will pick the arm with the best score $B_n^{(k)} := \hat{R}^{(k)}(n-1) + \Delta_n^{(k)}$ and a positive value of $I_n^{(k)}$ (line 9), and the threshold of that cycle is determined by the threshold function (line 10). A new cycle then starts according to this decision. When the cycle is finished, the meta-scheduler observes the (possibly clipped) cycle length and reward before updating the empirical rate for the selected arm. The updated statistics are then used to determine the next decision.

IV. Main Results and Discussion

Compared with previous works in the literature, our UCB-type algorithm tackles more challenging assumptions on cycle variables (sub-exponential instead of sub-Gaussian), and further the cycle lengths can be unbounded with positive probability, due to unstable schedulers. Thus, it requires several novel design choices such as dynamic interruption thresholds, a stability indicator, and a suitably modified exploration bonus. In this section, we will discuss these key design choices that differ from the classical UCB, followed by the result of the regret analysis.

A. Analysis of Key Design Choices

1) Hyper-Parameters: Before discussing the details of the meta-scheduler, let us first introduce several parameters used in our algorithm in the following assumption.

Assumption 3: We assume parameters $\mu_{\min}$, $\mu_{\max}$, $r_{\max}$ and (ν^2, α) are given a priori such that there exists a subset of arms $\mathcal{A}_0$ satisfying

$$\{k^*\} \subseteq \mathcal{A}_0 \subseteq \mathcal{A}^s(\boldsymbol{\lambda})$$

and for all $k \in \mathcal{A}_0$,

(1) $\mu_{\min} \le \mathbb{E}[C^{(k)}(1)] \le \mu_{\max}$.

(2) $\mathbb{E}[U^{(k)}(1) | C^{(k)}(1) = l] \le r_{\max} l$ for all $l \ge 1$. Note that $r_{\max}$ exists by Assumption 1, and $r_{\max} \ge r^{(*)}$.

(3) $C^{(k)}(1), U^{(k)}(1)$ are both (ν^2, α)-sub-exponential random variables as described in Assumption 2. In addition, we assume that the l-interrupted cycle reward $\hat{U}^{(k,l)}(1)$ is (ν^2, α)-sub-exponential for all $l \ge 2\mathbb{E}[C^{(k)}(1)]$.

In the algorithm, these parameters serve as hyper-parameters that need to be further tuned. To remove ambiguity, for a given set of hyper-parameters used in implementation, we will refer to $\mathcal{A}_0$ as the *largest* set of arms which satisfy these conditions with respect to those hyper-parameters. We assume $k^* \in \mathcal{A}_0$ (as the weakest notion) to achieve sub-linear regret.

Remark 5: For technical reasons, we also require $\hat{U}^{(k,l)}(1)$ to be sub-exponential under the same parameters[5] *(ν^2, α) as those of $U^{(k)}(1)$ when l is sufficiently large. This does not make the assumption significantly stronger, since one can always pick the parameters large enough to satisfy this condition. The condition $l \ge 2\mathbb{E}[C^{(k)}(1)]$ is chosen for simplicity. Indeed, the condition can be replaced by $l \ge (1+\gamma)\mathbb{E}[C^{(k)}(1)]$ for any $\gamma > 0$ (the algorithm parameters will be changed accordingly), which will be explained in Section IV-A.4.*

[5]Note that $\hat{U}^{(k,l)}(1)$ is sub-exponential (indeed bounded). However, the fact that $U^{(k)}(1)$ is (ν^2, α)-sub-exponential does not imply the same parameters suffice $\hat{U}^{(k,l)}(1)$.

2) Threshold Function: Recall that the interruption threshold for arm k's s-th sample is given by

$$f_s := \beta + \kappa \log s. \tag{9}$$

We require β and κ satisfy that

$$\beta > \mu_{\max} + \nu^2/\alpha, \quad \kappa/\alpha \ge 4.$$

Under these conditions, and by Assumption 2, it is easy to verify that the interruption probability for the best arm $\mathbb{P}(C_s^{(*)} > f_s^{(*)}) \le 1/s^2$. This implies that, under our threshold function, the expected number of interruptions of the best arm is bounded by a constant (since $\sum_{s=1}^{\infty} 1/s^2 = \pi^2/6$), and thus packet drops induced by the "wrong" interruptions do not grow faster than $O(\log n)$ over n cycles.

3) Stability Indicator: The stability indicator is defined as follows:

$$I_n^{(k)} = \begin{cases} 1 & \text{if } \sum_{i=1}^{T_{n-1}^{(k)}} \mathbb{1}\{C_i^{(k)} > f_i\} < \dfrac{\pi^2}{6} + \sqrt{2T_{n-1}^{(k)} \log n}, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

The Bernoulli random variable $\mathbb{1}\{C_s^{(k)} > f_s\}$ denotes whether the s-th sample for arm k is clipped by its threshold. The value $\mathbb{E}[\sum_{i=1}^{s} \mathbb{1}\{C_i^{(k)} > f_i\}]$ is bounded by $\pi^2/6$ for the optimal arm as previously discussed, but grows linearly for the unstable arms (since they are frequently clipped).

This motivates us to distinguish unstable arms by comparing the total number of interruptions of each arm to $\pi^2/6$ plus a concentration bound. By McDiarmid's inequality, the optimal arm's indicator $I_n^{(*)}$ is equal to 1 *w.h.p.* for any $n \ge 1$. In contrast, for any unstable k, the value of $\sum_{i=1}^{s} \mathbb{1}\{C_i^{(k)} > f_i\}$ will eventually exceed the limit.

4) Upper Confidence Bound: As discussed in the last section, an exploration bonus term is designed to compensate the empirical reward rates of arms for arm selection. Specifically, at least for the best arm, the empirical rate plus its exploration bonus should be above the true rate *w.h.p.*, which ensures the best arm gets sufficiently explored.

The exploration bonus is formally defined as follows:

$$\Delta_n^{(k)} = \Delta(\epsilon_n^{(k)}, \epsilon_n'^{(k)}) \tag{11}$$

where

$$\Delta(\epsilon, \epsilon') := \frac{\epsilon(1 + r_{\max}) + \epsilon'}{\mu_{\min} + \epsilon}, \tag{12}$$

and

$$\epsilon_n^{(k)} = \begin{cases} \sqrt{\dfrac{6\nu^2 \log n}{T_{n-1}^{(k)}}} & \sqrt{\dfrac{6\nu^2 \log n}{T_{n-1}^{(k)}}} \le \dfrac{\nu^2}{\alpha}, \\ \dfrac{6\alpha \log n}{T_{n-1}^{(k)}} & \text{otherwise,} \end{cases} \tag{13}$$

$$\epsilon_n'^{(k)} = \frac{1}{T_{n-1}^{(k)}} \sum_{i=1}^{T_{n-1}^{(k)}} \frac{r_{\max}}{i^{\kappa/2\alpha}} e^{-\beta/4\alpha} (\beta + 2\alpha + \kappa \log i + 1). \tag{14}$$

To see why this term works, first let us suppose every arm is stable and there is no interruption, i.e., $f_s = \infty$. As in [28],

we can introduce the following term,

$$\bar{\Delta}(\epsilon) := \frac{\epsilon(1+r_{\max})}{\mu_{\min}+\epsilon} \geq r^{(*)} - \frac{\mathbb{E}[U_1^{(*)}]-\epsilon}{\mathbb{E}[C_1^{(*)}]+\epsilon}. \tag{15}$$

By simple observation, we have that

$$\{\hat{R}_s^{(*)} + \bar{\Delta}(\epsilon) \leq r^{(*)}\}$$
$$\subset \{\frac{1}{s}\sum_{i=1}^{s}\hat{C}_i^{(*,f_i)} > \mathbb{E}[C_1^{(*)}]+\epsilon\} \bigcup \{\frac{1}{s}\sum_{i=1}^{s}\hat{U}_i^{(*,f_i)} \leq \mathbb{E}[U_1^{(*)}]-\epsilon\}. \tag{16}$$

These two events at the bottom allow us to use the concentration properties stated in Assumption 2 to bound the probability of the original event. Following a trick in [36], let $\epsilon = \epsilon_{n,s}$ where

$$\epsilon_{n,s} = \begin{cases} \sqrt{\dfrac{6\nu^2\log n}{s}} & \sqrt{\dfrac{6\nu^2\log n}{s}} \leq \dfrac{\nu^2}{\alpha}, \\ \dfrac{6\alpha\log n}{s} & \text{otherwise.} \end{cases} \tag{17}$$

We can then show that both of those two events happen with probability $1/n^3$ for all $s \leq n$. Hence, by taking a union bound on all possible $T_{n-1}^{(*)} \leq n$, we have that *w.h.p.* the exploration bonus $\bar{\Delta}(\epsilon_{n,T_{n-1}^{(*)}})$ suffice to compensate the best arm's empirical rate $\hat{R}^{(*)}(n)$ such that $\hat{R}^{(*)}(n)+\bar{\Delta}(\epsilon_{n,T_{n-1}^{(*)}}) > r^{(*)}$, which is as desired.

When we consider the threshold f_s as defined in (9), however, $\bar{\Delta}$ is not sufficient to compensate for $\hat{R}_s^{(*)}$ to exceed $r^{(*)}$. This is due to the bias of estimating $\mathbb{E}[U_1^{(*)}]$ by the average *truncated* reward $(1/s)\sum_{i=1}^{s}\hat{U}_i^{(*,f_i)}$, and the second bottom event in (16) is no longer with a negligible probability. This motivates us to adjust ϵ to account for the additional bias.

When $\beta > (1+\gamma)(\mu_{\max}+\nu^2/\alpha)$, by simple algebra, we have that the bias is bounded as follows (see the detailed derivation in Appendix A in the supplementary material).

$$\mathbb{E}[U_1^{(*)}] - \frac{1}{s}\sum_{i=1}^{s}\mathbb{E}[\hat{U}_i^{(*,f_i)}]$$
$$\leq \epsilon'_{n,s} := \frac{1}{s}\sum_{i=1}^{s}\frac{r_{\max}}{i^{\kappa/2\alpha}}(\beta+2\alpha+\kappa\log i+1)e^{-\beta\gamma/2(1+\gamma)\alpha}. \tag{18}$$

For simplicity, we let $\gamma = 1$ in our algorithm. Note that $\epsilon'_{n,s} \sim O(\log s/s)$ when $\kappa/\alpha \geq 4$.

Now we can define an updated exploration bonus $\Delta(\epsilon,\epsilon')$ as in (12). Note that

$$\Delta(\epsilon,\epsilon') := \frac{\epsilon(1+r_{\max})+\epsilon'}{\mu_{\min}+\epsilon} \geq r^{(*)} - \frac{(\mathbb{E}[U_1^{(*)}]-\epsilon')-\epsilon}{\mathbb{E}[C_1^{(*)}]+\epsilon}. \tag{19}$$

Following the same logic as in (16), we can conclude[6] that *w.h.p.*, $\hat{R}^{(*)}(n) + \Delta(\epsilon_{n,T_{n-1}^{(k)}}, \epsilon'_{n,T_{n-1}^{(k)}}) > r^{(*)}$.

As a sanity check, if $T_n^{(k)}$ grows faster than $O(\log n)$ for any stable arm k, then its exploration bonus will converge to 0. Thus, the UCB-compensated empirical rate of a stable suboptimal arm will eventually fall short of $r^{(*)}$ after sufficient explorations.

[6]The proof requires the technical assumption in item (3) of Assumption 3.

B. Main Results

In this part we present the main theorem, which justifies that the meta-scheduler given in Algorithm 1 satisfies our requirements regarding negligible packet loss and a sub-linear regret. Before that, let us first introduce a key lemma, stating that the number of sub-optimal decisions under our algorithm grows quasi-logarithmically.

For simplicity, let $d^{(k)} := r^{(*)} - r^{(k)}$ denote the gap of reward rates between arm k and the optimal arm. We will use symbols $\wedge$ and $\vee$ as the shorthand notations for $\min$ and $\max$ functions respectively.

Lemma 1: A meta-scheduler implementing Algorithm 1 satisfies the following regarding $\mathbb{E}[T_n^{(k)}]$.

(1) For all unstable arms $k \in \mathcal{A}^u(\boldsymbol{\lambda})$,

$$\mathbb{E}[T_n^{(k)}] \leq \lceil \mathsf{K}_1^{(k)}\log n\rceil + \frac{\pi^2}{6} + 1$$

where

$$\mathsf{K}_1^{(k)} = \frac{18}{\mathbb{P}(C_1^{(k)}=\infty)^2}.$$

(2) For stable suboptimal arms that lie in set $\mathcal{A}_0 \setminus \{k^*\}$, we have that

$$\mathbb{E}[T_n^{(k)}] \leq \lceil \mathsf{K}_2^{(k)}\log n \vee \mathsf{M}_1^{(k)} \vee \mathsf{M}_2^{(k)}\rceil + \pi^2 + 1$$

where

$$\mathsf{K}_2^{(k)} = \frac{24(1+r_{\max})^2\nu^2(\mu_{\min}+1)^2}{(d^{(k)})^2\mu_{\min}^2} \vee \frac{12(1+r_{\max})\alpha(\mu_{\min}+1)}{d^{(k)}\mu_{\min}}$$

and $\mathsf{M}_1^{(k)}, \mathsf{M}_2^{(k)}$ are (smallest possible) constants such that

$$\mathsf{M}_1^{(k)} \geq \frac{4}{d^{(k)}}\bar{r}(\mathbb{E}[C_1^{(k)}]+6\alpha_k\log\mathsf{M}_1^{(k)})(\frac{\pi^2}{6}+\sqrt{\mathsf{M}_1^{(k)}\log\mathsf{M}_1^{(k)}}),$$

$$\mathsf{M}_2^{(k)} \geq \frac{4}{d^{(k)}}r_{\max}\cdot\frac{\pi^2}{6}e^{-\beta/4\alpha}(\beta+2\alpha+\kappa\log\mathsf{M}_2^{(k)}+1).$$

(3) For stable suboptimal arms that lie in $\mathcal{A}^s(\boldsymbol{\lambda}) \setminus \mathcal{A}_0$, we have that for any $\delta > 0$ and $\chi > 1$,

$$\mathbb{E}[T_n^{(k)}] \leq \lceil \mathsf{K}_3^{(k)}\log n \vee \mathsf{J}_1^{(k)}\log^{1+\delta} n \vee \chi \vee \mathsf{M}_2^{(k)}\rceil + \pi^2 + 1$$

where

$$\mathsf{K}_3^{(k)} = \frac{24(1+r_{\max})^2\nu_k^2(\mu_{\min}+1)^2}{(d^{(k)})^2\mu_{\min}^2} \vee \frac{12(1+r_{\max})\alpha_k(\mu_{\min}+1)}{d^{(k)}\mu_{\min}}$$

and $\mathsf{J}_1^{(k)}$ is a constant (up to δ and χ) such that we have the equation at the bottom of the next page.

Note that $\mathsf{M}_1^{(k)}, \mathsf{J}_1^{(k)}$ are roughly $O((1/d^{(k)})^2)$ and $\mathsf{M}_2^{(k)}$ is roughly $O(1/d^{(k)})$. To summarize,

$$\mathbb{E}[T_n^{(k)}] = \begin{cases} O(\log n) & \forall k \in \mathcal{A}^u(\boldsymbol{\lambda}), \\ O(\log n) & \forall k \in \mathcal{A}_0 \setminus \{k^*\}, \\ O(\log^{1+\delta} n) & \forall \delta > 0 \ \ \forall k \in \mathcal{A}^s(\boldsymbol{\lambda}) \setminus \mathcal{A}_0. \end{cases}$$

Proof of Lemma 1: Here we will present a proof sketch. The complete proof can be found Appendix B in the supplementary material.

The proof consists of two parts. First, we prove the case when arm k is unstable. Observe that for any $k \in \mathcal{A}^u(\boldsymbol{\lambda})$, $p_k := \mathbb{P}(C_1^{(k)}=\infty) > 0$. Thus, *w.h.p.*, the value $\sum_{i=1}^{T_{n-1}^{(k)}}\mathbb{1}\{C_i^{(k)} > f_i\}$ grows no slower than $p_k T_{n-1}^{(k)}$ minus a concentration bound $\sqrt{2T_{n-1}^{(k)}\log n}$ (by Bernstein's inequality). Therefore, if the n-th arm decision $A_n = k \in \mathcal{A}^u(\boldsymbol{\lambda})$, by the stability indicator, we have that $p_k T_{n-1}^{(k)} - \sqrt{2T_{n-1}^{(k)}\log n} < \pi^2/6 + \sqrt{2T_{n-1}^{(k)}\log n}$. This implies $\mathbb{E}[T_n^{(k)}] = O(\log n)$.

Next, we study the case for any stable suboptimal arm. If $A_n = k \in \mathcal{A}^s(\boldsymbol{\lambda}) \setminus \{k^*\}$, either of the following two events will happen: (a) the stability indicator of the best arm is 0; or (b) the UCB score (empirical rate plus UCB) of arm k is larger than the best arm's score. As discussed in IV-A.3, the first event has a negligible chance to happen. For the second event, we already guarantee that *w.h.p.*, $\hat{R}^{(*)}(n) + \Delta(\epsilon_n^{(*)}, \epsilon_n'^{(*)}) > r^{(*)}$ by the design of the exploration bonus in IV-A.4. Then to show the second event cannot occur either, it suffices to show that $\hat{R}^{(k)}(n) + \Delta(\epsilon_n^{(k)}, \epsilon_n'^{(k)}) < r^{(*)}$. Observe that $\hat{R}^{(k)}(n) \approx r^{(k)} < r^{(*)}$ and that $\Delta(\epsilon_n^{(k)}, \epsilon_n'^{(k)})$ can be arbitrarily small when $T_n^{(k)} > \mathsf{C} \log n$ for a sufficiently large C. This suggests $T_n^{(k)}$ cannot grow faster than $O(\log n)$. When $k \in \mathcal{A}^s(\boldsymbol{\lambda}) \setminus \mathcal{A}_0$, an additional $1+\delta$ is needed in the exponent of $\log n$. This is caused by a technical issue[7] due to the bias of (possibly) overestimating $r^{(k)}$ by $\hat{R}^{(k)}(n)$. □

Now we present the main theorem as follows.

Theorem 1: Let $f_\tau = (\kappa + \alpha \log \tau)$ *and*

$$\mathsf{g}_1^{(k)}(\tau) = \lceil \mathsf{K}_1^{(k)} \log \tau \rceil + 1,$$
$$\mathsf{g}_2^{(k)}(\tau) = \lceil \mathsf{K}_2^{(k)} \log \tau \vee \mathsf{M}_1^{(k)} \vee \mathsf{M}_2^{(k)} \rceil + 1,$$
$$\mathsf{g}_3^{(k)}(\tau) = \lceil \mathsf{K}_3^{(k)} \log \tau \vee \mathsf{J}_1^{(k)} \log^{1+\delta} \tau \vee \chi \vee \mathsf{M}_2^{(k)} \rceil + 1,$$

where $\mathsf{K}_1^{(k)}$, $\mathsf{K}_2^{(k)}, \mathsf{K}_3^{(k)}, \mathsf{J}_1^{(k)}, \mathsf{M}_1^{(k)}$ *and* $\mathsf{M}_2^{(k)}$ *are defined as in Lemma 1. The meta-policy* π *induced by Algorithm 1 has the following properties.*

(1) For the expected number of packets discarded over the time horizon τ*, denoted by* $\mathbb{E}[D_\pi[\tau]]$*, we have for any* $\delta > 0$ *and* $\chi > 1$*,*

$$\begin{aligned}\mathbb{E}[D_\pi[\tau]] &\le \bar{a} u f_\tau \Bigg(\sum_{k \in \mathcal{A}_0} \frac{\pi^2}{6} \\ &+ \sum_{k \in \mathcal{A}^u(\boldsymbol{\lambda})} \left(((\frac{\pi^2}{6} + \sqrt{2\mathsf{g}_1^{(k)}(\tau) \log \tau} + 2) \wedge \mathsf{g}_1^{(k)}(\tau)) + \frac{\pi^2}{6} \right) \\ &+ \sum_{k \in \mathcal{A}^s(\boldsymbol{\lambda}) \setminus \mathcal{A}_0} \left(((\frac{\pi^2}{6} + \sqrt{2\mathsf{g}_3^{(k)}(\tau) \log \tau} + 2) \wedge \mathsf{g}_3^{(k)}(\tau)) + \pi^2 \right) \Bigg).\end{aligned}$$

(2) For the regret $\mathrm{Reg}_\pi[\tau]$*, we have for any* $\delta > 0$ *and* $\chi > 1$*,*

$$\begin{aligned}\mathrm{Reg}_\pi[\tau] \le &\sum_{k \in \mathcal{A}^u(\boldsymbol{\lambda})} (r^{(*)} - \tilde{r}^{(k)})(\mathsf{g}_1^{(k)}(\tau) + \frac{\pi^2}{6}) f_\tau \\ &+ \sum_{k \in \mathcal{A}^s(\boldsymbol{\lambda}) \setminus \mathcal{A}_0} (r^{(*)} - \tilde{r}^{(k)})(\mathsf{g}_3^{(k)}(\tau) + \pi^2) f_\tau \\ &+ \sum_{k \in \mathcal{A}_0} (r^{(*)} - r^{(k)})(\mathsf{g}_2^{(k)}(\tau) + \pi^2) \mu_{\max} \\ &+ r^{(*)} f_\tau + r^{(*)} \frac{\mathbb{E}[(C^{(*)}(1))^2]}{\mathbb{E}[C^{(*)}(1)]} + \sum_{k \in \mathcal{A}_0} \mathsf{h}(\tau),\end{aligned}$$

where

$$\tilde{r}^{(k)} = \inf_{l \ge \beta} \frac{\mathbb{E}[\hat{U}^{(k,l)}(1)]}{\mathbb{E}[\hat{C}^{(k,l)}(1)]},$$

$$\mathsf{h}(\tau) = r_{\max} \frac{\pi^2}{6} e^{-\beta/4\alpha} (f_\tau + 2\alpha + 1).$$

This theorem implies that if $\mathcal{A}_0 = \mathcal{A}^s(\boldsymbol{\lambda})$, i.e., all the stable arms satisfy the conditions in Assumption 3 with respect to the hyper-parameters used in the algorithm, then

$$\mathbb{E}[D_\pi[\tau]] = O(\log^2(\tau)), \quad \mathrm{Reg}_\pi[\tau] = O(\log^2 \tau).$$

Otherwise,

$$\mathbb{E}[D_\pi[\tau]] = O(\log^{2+\delta}(\tau)), \quad \mathrm{Reg}_\pi[\tau] = O(\log^{2+\delta}(\tau)), \quad \forall \delta > 0.$$

Proof of Theorem 1: The complete proof can be found Appendix C in the supplementary material.

Claim (1) of the theorem: The expected number of interruptions on arms in $\mathcal{A}_0$ is bounded by $\pi^2/6$ due to the threshold design. For any arm not in $\mathcal{A}_0$, the expected number of interruptions is bounded by the number of arm selections $\mathbb{E}[T_\tau^{(k)}]$ (note that $N_\pi[\tau] \le \tau$), but a nicer bound can be given since some of the cycles might not be interrupted. For each interrupted cycle, the number of packets dropped is bounded by $\bar{a} u f_\tau = O(\log \tau)$, where f_τ is a (coarse) bound on the longest possible cycle before time τ. This concludes the claim.

Claim (2) of the theorem: For this part we follow a similar approach as in the budgeted bandit literature [26], [28]. We first bound the number of sub-optimal actions that have been taken in order to bound regret. Intuitively, the expected cumulative reward for the optimal meta-policy is roughly $r^{(*)}\tau$, and the regret of π is due to the reward loss during the period of suboptimal decisions, where the loss in reward rate is either $r^{(*)} - r^{(k)}$ (for $k \in \mathcal{A}_0 \setminus \{k^*\}$) or bounded by $r^{(*)} - \tilde{r}^{(k)}$ (for $k \notin \mathcal{A}_0$).[8] We can show that

$$\begin{aligned}\mathrm{Reg}[\tau] \le &\sum_{k \in \mathcal{A}_0} \mathbb{E}[T_\tau^{(k)}] \mu_{\max} (r^{(*)} - r^{(k)}) \\ &+ \sum_{k \notin \mathcal{A}_0} \mathbb{E}[T_\tau^{(k)}] f_\tau (r^{(*)} - \tilde{r}^{(k)}) + O(\log \tau).\end{aligned}$$

The value of $\mathbb{E}[T_\tau^{(k)}]$ is then bounded using Lemma 1. □

As described in Remark 4, if packet dropping is unacceptable in the system, one can instead switch to a queue-stabilizing (throughput-optimal) policy like MaxWeight to clear out the queues. This process introduces a small extra cost to the total regret.

Corollary 1: Once a cycle is interrupted, if the meta-scheduler switches to a MaxWeight (instead of dropping

[7] If the hyper-parameters, in particular β and α, do not suffice the conditions in Assumption 3 for a stable suboptimal arm k, the number of interruptions of k is no longer well bounded under our threshold function. This leads to an additional bias between $r^{(k)}$ and $\hat{R}^{(k)}(n)$.

[8] When $k \in \mathcal{A}_0$, the expected reward rate of arm k's period (which may include interrupted cycles) is roughly $r^{(k)}$, since only few cycles are interrupted; otherwise, we use a weaker bound $\tilde{r}^{(k)}$ instead.

$$\mathsf{J}_1^{(k)} \ge \frac{4}{d^{(k)} \bar{r}} \frac{(\mathbb{E}[C_1^{(k)}] + 6\alpha_k \log(\mathsf{J}_1^{(k)} \log^{1+\delta} \chi))(\frac{\pi^2}{6} + \sqrt{2\mathsf{J}_1^{(k)} \log^{2+\delta} \chi} + 1)}{\log^{1+\delta} \chi}.$$

Algorithm 2 UCB Meta-Scheduler With Interruption for System With Constraints

Input: Set of scheduling policies $\mathcal{A}$.
Threshold Function: $f_s := \beta + \kappa \log s$.
Initialization: Run every arm k once with interruption threshold β, then initialize empirical rates $\hat{R}_1^{(k)}$ and $\hat{W}_{i,1}^{(k)}, \forall i = 1, 2, \cdots, h$.
for $n = |\mathcal{A}| + 1, |\mathcal{A}| + 2, \cdots$ **do**
  [*before cycle* n]
  for all $k \in \mathcal{A}$ **do**
    Compute *Exploration Bonus* $\Delta_n^{(k)}$.
    Compute *Stability Indicator* $I_n^{(k)}$.
      ▷ $\Delta_n^{(k)}$ and $I_n^{(k)}$ as defined in Section IV.
    Compute *Constraint Indicator* $J_n^{(k)}$.
  Arm decision:
    $A_n \in \underset{k \in \mathcal{A}}{\operatorname{argmax}}\ I_n^{(k)} \times J_n^{(k)} \times (\hat{R}^{(k)}(n-1) + \Delta_n^{(k)})$.
  Cycle interruption decision: $L_n = f_{T_n^{(A_n)}}$.
  [*after cycle* n]
  Observe $\hat{C}^{(A_n, L_n)}(n), \hat{U}^{(A_n, L_n)}(n)$ and $\hat{V}_i^{(A_n, L_n)}(n)$.
  Update $\hat{R}^{(A_n)}(n)$ and $\hat{W}_i^{(A_n)}(n), \forall i$.

packets) until the system returns idle, the total regret $\mathrm{Reg}_\pi[\tau]$ *satisfies the following*:

$$\mathrm{Reg}_\pi[\tau] = \begin{cases} O(\log^3 \tau) & \text{if } \mathcal{A}_0 = \mathcal{A}^s(\boldsymbol{\lambda}), \\ O(\log^{3+\delta} \tau),\ \forall \delta > 0 & \text{otherwise.} \end{cases}$$

Proof: By Lyapunov stability, it can be shown that the average time required for MaxWeight to clear out queues of a total length q is in the order of $O(q^2)$. Recall that when a cycle is interrupted before the horizon τ, the total queue length is bounded by $\mathsf{C} \cdot \log \tau$ for some constant C. Therefore, it takes $O(\log^2 \tau)$ time slots for each queue-clearing process to end, and the extra regret induced is thereby in the order of $O(r^{(*)} \cdot \log^2 \tau)$ since $r^{(*)}$ is the rate of the optimal policy. Then the claim is shown by utilizing Lemma 1 and combining with Theorem 1. A complete proof is presented Appendix D in the supplementary material. □

V. Extension: System With Constraints

In the previous sections, we introduced a UCB-type meta-scheduler that determines the best stable policy optimizing the renewal reward rate of the system. In some applications, the system might be also interested in satisfying a certain performance guarantee besides maximizing the main reward. For instance, the system may attempt to minimize mean packet delay of all traffic but also promise that certain users get sufficient service (*e.g.*, 80% of packets must arrive within 5 ms). If the guarantee can be described as a constraint on the reward rate of another renewal-reward process (other than the main reward), we can extend Algorithm 1 to locate the optimal constraint-satisfying policy with a simple modification.

First let us generalize the basic model as follows. For arm k, the n-th cycle $C^{(k)}(n)$ is associated with 1 cycle reward $U^{(k)}(n)$ and h auxiliary rewards $V_1^{(k)}(n), \cdots, V_h^{(k)}(n)$. Both the main and auxiliary rewards satisfy Assumption 1. If $\pi_n = (k, l)$, a stochastic feedback Z_n is observed for n-th cycle as follows,

$$Z_n = (\hat{C}^{(k,l)}(n), \hat{U}^{(k,l)}(n), \hat{V}_1^{(k,l)}(n), \cdots, \hat{V}_h^{(k,l)}(n), \mathbb{1}\{C^{(k)}(n) > l\}).$$

As in (4), let $w_i^{(k)} := \mathbb{E}[V_i^{(k)}(1)]/\mathbb{E}[C^{(k)}(1)]$ be the renewal reward rate for i-th auxiliary reward of arm k. We call a scheduling policy *acceptable* if it guarantees that the reward rates for the h auxiliary rewards exceed a given threshold $\boldsymbol{\xi} = (\xi_1, \cdots, \xi_h)$, which is known a priori by the meta-scheduler. The optimal arm k^* is thus defined as

$$k^* = \underset{k}{\operatorname{argmax}}\ r^{(k)} \quad \text{s.t.}\ \ k \in \mathcal{A}^s \cap \{k' | w_i^{(k')} \geq \xi_i, \forall i = 1, 2, \cdots, h\}.$$

Inspired by the stability indicator, the constraints can also be handled by an indicator that eliminates unacceptable arms with high probability. Since auxiliary rewards still satisfy Assumption 1, we can use the same UCB bound as defined in (19), and an arm's *constraint indicator* is set to be false when its empirical rate of auxiliary rewards compensated by the exploration bonus is below $\boldsymbol{\xi}$. Denote $\hat{W}_i^{(k)}(n)$ as the (observed) empirical rate (see (8)) of i-th auxiliary reward after n cycles. Formally, the constraint indicator $J_n^{(k)}$ is as follows,

$$J_n^{(k)} = \prod_{i=1}^{h} \mathbb{1}_{\{\hat{W}_i^{(k)}(n-1) + \Delta_n^{(k)} \geq \xi_i\}}.$$

The algorithm is formally presented in Algorithm 2.

VI. Performance Evaluation

In this section, we evaluate the performance of our meta-scheduler algorithms. The simulation setting is based on the IMT Advanced evaluation guidelines for urban macro-cell deployments [37]. We consider a wireless network consisting of a single base station (BS) and $u = 12$ down-link users. The BS is located at the center of the cell with a radius of 250 m, and the user terminals are located in the cell. We assume the total channel bandwidth is 10 MHz. Further, the bandwidth can be divided into 200 resource units of 0.05 MHz each, which can be assigned to different users within a time slot. Scheduling decisions, which consist of the allocation of each resource unit, are made once in each time slot of duration 0.5 ms.

We assume that the size of packets in the system is fixed at 5 kb. At each slot, each users' packets arrive as *i.i.d.* binomial random variables. For simplicity we do not allow one packet to be transmitted across several time slots. The user scheduling within one slot is done in a sequential manner: one of the users is first scheduled for 1 packet based on current queue and rate values, then the updated queues and rates (of the remaining resource units) are used to determine the next user. The process is iterated until remaining resource units cannot support another packet transmission.

The Signal-to-Interference-Noise ratio (SINR) at time t is modeled as $\mathtt{SNR}_i[t] = P_b g_i[t]/(\sigma^2 + I_i[t])$ where P_b is the transmit power of BS, $g_i[t]$ denotes the channel gain of user i, σ^2 and $I_i[t]$ denote the noise and the interference level respectively. The channel gain is a combination of path loss, fast fading and antenna gain, Following [37], we set $P_b = 47$ dBm, $\sigma^2 = -104$ dBm, path loss (in dB) computed as $39.1 \log_{10}(\mathtt{dist}) + 13.5 + 20 \log_{10}(f_c)$ where

TABLE I

A SUMMARY OF POLICIES USED IN OUR SIMULATIONS. HERE WE OMIT ALL TIME INDICES, AND Q_i, H_i, S_i DENOTE THE CURRENT QUEUE LENGTH, HEAD-OF-LINE DELAY AND AVAILABLE SERVICE RATE OF USER i **AT THE TIME OF DECISION FOR EACH PACKET** (REMIND THAT USER SCHEDULING IS DONE PACKET BY PACKET SEQUENTIALLY IN A TIME SLOT). UNLESS SPECIFICALLY MENTIONED WE WILL SET POLICY PARAMETERS AS BELOW. NOTE THAT THE POLICIES ARE ALL WEIGHTED (I.E., b_i) BY THE INVERSE OF MEAN RATE OF EACH USER, WHICH IS A COMMON PRACTICE SUGGESTED IN [11]

	Policy Rules	Parameter Settings
MaxWeight	$i^* = \text{argmax}_{i\in\mathcal{U}}\, b_i S_i Q_i$	$b_i = 1/\mathbb{E}[S_i]$
Exp-Rule	$i^* = \text{argmax}_{i\in\mathcal{U}}\, b_i S_i \exp\left(\frac{a_i Q_i}{c+(u^{-1}\sum_{j\in\mathcal{U}} a_j Q_j)^\eta}\right)$	$b_i = 1/\mathbb{E}[S_i], a_i = 1, c = 0.3, \eta = 0.6$
Log-Rule	$i^* = \text{argmax}_{i\in\mathcal{U}}\, b_i S_i \log(c + a_i Q_i)$	$b_i = 1/\mathbb{E}[S_i], a_i = 2, c = 1$
MaxWeight-HOL	$i^* = \text{argmax}_{i\in\mathcal{U}}\, b_i S_i H_i$	$b_i = 1/\mathbb{E}[S_i]$
Max-Rate	$i^* = \text{argmax}_{i\in\mathcal{U}}\, b_i S_i$	$b_i = 1/\mathbb{E}[S_i]$

TABLE II

USER PROFILE IN OUR SIMULATION SYSTEM

User Indices	1,2	3,4	5,6	7,8	9,10	11,12
Distance to BS (m)	50	80	110	140	170	200
Mean Rate (packets/slot)	9.16	6.54	4.84	3.62	2.71	2.04

$f_c = 2.0$ GHz and `dist` denotes the user distance, and antenna gain of 17 dBi. Fast fading follows a Rayleigh distribution and is independent over users. For simplicity, we assume the interference is identical to all users and $I_i[t] = -56$ dBm. In any time slot t, the channel state (rate supported by the channel) of the user i is given by

$$S_i[t] = \texttt{BW} \times \log_2(1 + 10^{0.1(\texttt{SNR}_i[t]-\texttt{L})}) \quad \text{bps}$$

where the parameter $\texttt{L} = 3$dB describes a loss to Shannon capacity.

In the following simulations, we will fix the locations of 12 users. The location profile and the mean data rates are given in Table II. Several classical scheduling policies we use are summarized in Table I.

A. Meta-Scheduler Behavior and Reward Design

In this experiment we select various types of rewards and show that the meta-scheduler can indeed pick the optimal policy. We set *i.i.d.* random arrival $A_i[t] \sim \text{Binomial}(3, 0.12)$ for each $i \in \mathcal{U}$ described in Table II. Under this arrival rate ($\lambda_i = 0.36$ packets/slot), cycle lengths induced by the policies in Table I are no more than 60 ms.

Suppose each packet of the system is associated with a reward and the cycle reward is simply defined as the sum of all packet rewards with proper normalization such that $r_{\max} = 1$ (see Assumption 3). Three types of packet rewards are considered as follows:

Type-1: Mean delay: The reward of each packet equals $(1-\texttt{delay} * 0.1)^+$. To optimize this type of reward is equivalent to minimize the mean delay of packets provided the delays are smaller than 10 time slots.

Type-2: Deadline requirement: Each packet receives a reward of '1' only if its delay is less than `ddl` slots. Otherwise the packet receives '0' reward. We use $\texttt{ddl} = 8$ in this experiment.

Type-3: Burstiness: This reward favors spreading the service allocations to a user across slots rather than serving a user multiple packets in a single slot. If a user receives a single packet within one slot, this packet is associated with a reward of '1'. If two or more packets are received in the same slot, it will be considered as "bursty" and no rewards are given to any of the packets.

TABLE III

A SUMMARY OF MEAN CYCLE LENGTHS AND REWARD RATES FOR 3 TYPES OF REWARDS CONSIDERED INDUCED BY EACH POLICY USED IN SECTION VI-A. THE REWARD RATE OF THE OPTIMAL ARM FOR EACH REWARD TYPE IS IN BOLD FONT

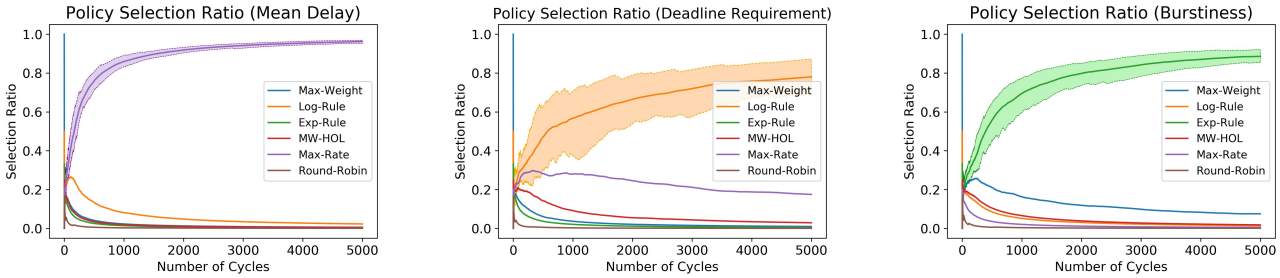
	Mean Cycle Length (ms)	Average Reward per Packet		
		Type 1	Type 2	Type 3
MaxWeight	37	0.717	0.887	0.589
Log-Rule	22	0.805	**0.954**	0.557
Exp-Rule	57	0.627	0.796	**0.599**
MW-HOL	48	0.694	0.931	0.567
Max-Rate	20	**0.832**	0.949	0.484
Round-Robin	$+\infty$	N/A	N/A	N/A

Besides the policies given in Table I, we also consider a Round-Robin scheduler as a baseline which may not be stable even if the traffic loads are within the capacity region. In Table III, we list the average cycle length and reward rates induced by each policy. We set the parameters of Algorithm 1 as $\alpha = 4, \nu^2 = 1, \kappa = 50, \beta = 200, \mu_{\min} = 20, r_{\max} = 1$ and run 40 simulations for each type of rewards. Define the selection ratio of arm k after n cycles as $(1/n)\sum_{i=1}^{n} \mathbb{1}\{A_i = k\}$. Figure 2 exhibits the mean selection ratios of all arms for the three types of rewards (with 10% and 90% quantile shown for the best arms). In each case, Algorithm 1 correctly determines the optimal policy. We observe the rate of convergence largely depends on the performance gap between the best and second best arms: Type-2 reward takes the longest time to separate between Log-Rule and Max-Rate since they have the least gap. As we would expect, Round-Robin scheduler gets discarded quickly in all cases.

B. Meta-Scheduler Behavior Dependence on the Load

In this experiment, we show the robustness of Algorithm 1 over variations in the traffic load. We design a case where the best policy shifts from one to another when we adjust the load of the system. The goal is to verify the optimal arm is picked by our algorithm in all scenarios.

Suppose Users 6, 12 are two reward Type-2 users with $\texttt{ddl} = 2$ (as defined in the last experiment) that are quite strict with packet deadlines, while other users are Type-1 users. The cycle reward is still defined as the sum of packet rewards of all



(a) Mean Delay (b) Deadline Requirement (c) Burstiness

Fig. 2. Mean policy selection ratio of Algorithm 1 for 3 types of rewards defined in Section VI-A, after simulating 40 times in each case. The area between 10% and 90% quantile of the best arm is shaded.

TABLE IV

A SUMMARY OF MEAN RENEWAL REWARD RATES FOR THE MAIN AND AUXILIARY REWARDS INDUCED BY EACH POLICY USED IN SECTION VI-C. ONLY THE LAST POLICY IS ACCEPTABLE WHEN $\boldsymbol{\xi} = [0.75, 0.75]$

	Average Reward per Packet		
	User 6	User 12	Others (Main Reward)
$\gamma = 0.8$	0.855	0.577	**0.808**
$\gamma = 1.0$	0.810	0.725	0.803
$\gamma = 1.2$	**0.772**	**0.834**	0.787

users. We consider two policies: 1) Log-Rule and 2) Priority Rule. The second policy is defined as follows: at each slot, Type-2 users are always scheduled first using a Max-Rate policy; Type-1 users are scheduled using Log-Rule only when there are no more Type-2 packets that can be transmitted. Clearly, the second policy provides better performance for Type-2 users.

We consider a system where each user has random arrivals $A_i[t] \sim \text{Binomial}(3, \lambda_i/3)$. We increase the traffic load from $\lambda_i = 0.32$ to 0.36 for each $i \in \mathcal{U}$. Figure 3a shows the reward per packet under the two policies as a function of the traffic load. When the load is relatively light, the priority-based scheduler outperforms Log-Rule; however, when the load is larger than 0.34, the reward boost of Priority Rule for Type-2 users does not compensate the loss in mean delay for Type-1 users and Log-Rule prevails instead. Indeed, Priority Rule is not even stable for even higher loads (see Figure 3b).

Figure 3d to 3f exhibit the simulation results for $\lambda_i = 0.32, 0.34$ and 0.36. Algorithm 1 correctly locates the optimal policy in the low and high load scenario. When $\lambda_i = 0.34$, the selection ratio of two policies barely separate as the performance gap is almost 0. This is not an issue for any MAB algorithm as both arms can be viewed as the best arm in this scenario.

Remark 6: Figure 3d-3f illustrate each arm's selection ratio over the number of cycles. Indeed, the meta-scheduler's rate of convergence ***in real time scale*** *also depends on cycle lengths. In general, two factors affect the rate of convergence. First, the larger is the performance gap between the best and second best arm, the easier it is to learn. Thus, as load increases, it is indeed possible that the instance becomes easier due to the increased gap between the best and second best schedulers. Second, the longer is the system's cycle time, the slower the learning process is. With this effect, in general, the system with higher loads will exhibit longer cycle times. To get some insight on the load-cycle relation, consider for simplicity a standard M/M/1 queue with $\bar{\lambda}$ and $\bar{\mu}$ as the mean arrival and service rate respectively. For a stable queue, we have the load parameter $\rho := \bar{\lambda}/\bar{\mu} < 1$. From standard analysis of such queues, the mean cycle length is $\bar{\mu}/(1-\rho) + 1/\bar{\lambda}$ and the sub-exponential parameter α roughly scales as $O(1/\log \rho^{-1}) \approx O(\rho/(1-\rho))$. Recall that the regret scales linearly in these parameters, and thus, the regret has an inverse dependence in $(1-\rho)$, assuming the performance gap is fixed. The system we consider is more complex, and includes opportunism, multi-user scheduling and a non-stationary schedule, thus making it hard to analytically quantify the effect.*

In Figure 3g, we numerically explore how the regret varies with load and indeed see a mixed impact – as the load increases, the regret does not change monotonically due to the different effects of enlarging performance gaps and growing cycle lengths.

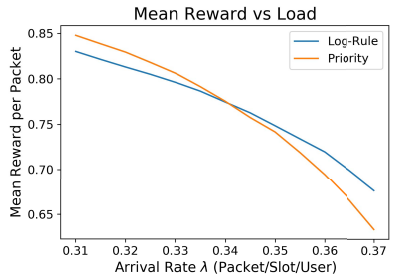
C. Meta-Scheduler Behavior With Performance Constraints

In this experiment, we consider the case where additional constraints are imposed on the system. Let $\lambda_i = 0.36$ packet per slot for any $i \in \mathcal{U}$. Let User 6 and User 12 be Type-2 users. Suppose we impose the following performance guarantee: 75% of packets for user 6 and 12 must arrive with a delay less than 5 slots ($\texttt{ddl} = 5$). And the target is to pick the policy that minimizes the mean delay of the other 10 users while satisfying this constraint.

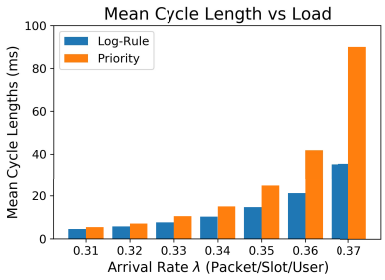
We are given 3 Log-Rule schedulers with different weight parameters b_i (See Table I): $b_i = \frac{1}{\mathbb{E}[S_i]^\gamma}$ where $\gamma = 0.8, 1$ and 1.2. Here γ roughly tunes the fairness of each user, and a larger γ is good for users with low average rates. Table IV summarizes the reward rates for the constraints and main objective. Only the policy with $\gamma = 1.2$ satisfies the constraints.

We run Algorithm 2 40 times using the same parameters as in the first experiment. Figure 4a shows the policy selection ratio over number of cycles with the constraints described above. As a comparison, we drop the constraint (by setting $\boldsymbol{\xi} = 0$) and the result is shown in Figure 4b. In both cases, Algorithm 2 locates the best constraint-satisfying policy.

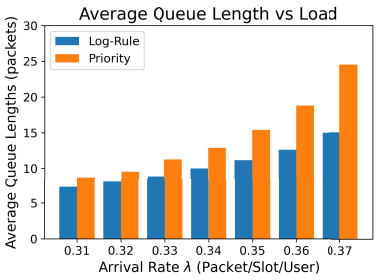
To clearly show the behavior of Algorithm 2, we manually slow the convergence of learning by increasing hyperparameter α from 4 to 20, which corresponds to a more conservative upper confidence bound. As shown in Figure 4c, the third policy prevails the selection ratio after the other two policies sequentially get dropped by the constraint indicators.



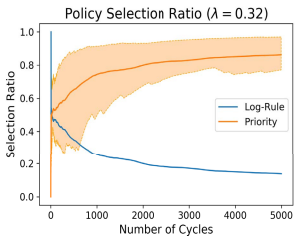
(a) Load vs Reward Rate.



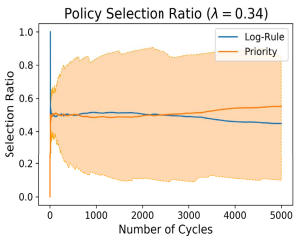
(b) Load vs Mean Cycle Length.



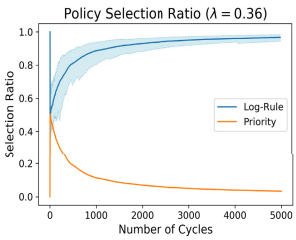
(c) Load vs Mean Sum-Queue Length.



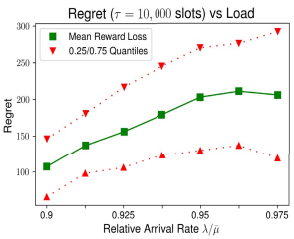
(d) $\lambda_i = 0.32$



(e) $\lambda_i = 0.34$

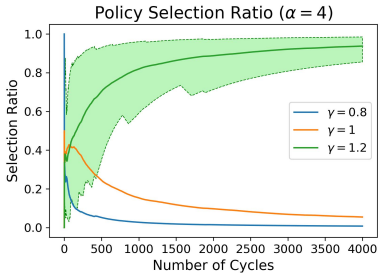


(f) $\lambda_i = 0.36$

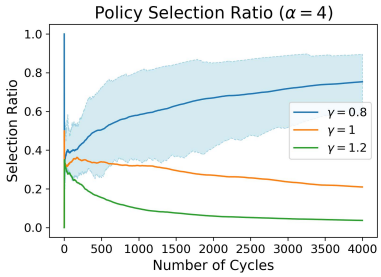


(g) Regret vs Load.

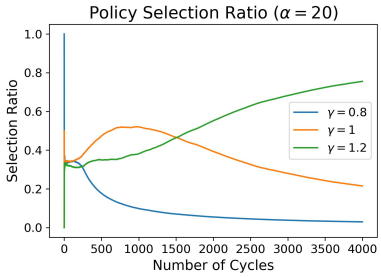
Fig. 3. Results for the experiment in Section VI-B. (a-c) Reward rate, mean cycle length and mean queue length induced by Log-Rule and Priority under changing traffic loads. (d-f) Mean policy selection ratio (40 simulations) of the meta-scheduler when arrival rate $\lambda_i = 0.32, 0.34, 0.36$ respectively, where the area between 10% and 90% quantile of the best arm is shaded. (g) Mean reward loss (aka regret) of meta-scheduler π for time horizon $\tau = 10$k over varying traffic loads. Let $\bar{\mu}$ be the mean (non-opportunistic) service rate and $\bar{\mu} = 0.4$ packet/user/slot. We focus on the high load region of $\lambda = 0.36$ to 0.39 (i.e., relative arrival rate > 0.9) where the best policy is Log-Rule. As the load increases, the expanding performance gap of two policies and the growing cycle lengths have opposing effects on the regret, and thus it does not grow monotonically.



(a) System with a constraint



(b) System without constraints



(c) System with a constraint (slow convergence)

Fig. 4. Results for the experiment in Section VI-C using Algorithm 2. (a) The meta-scheduler finds the best policy ($\gamma = 1.2$) subject to the performance constraint defined in Section VI-C. (b) The meta-scheduler finds the best policy ($\gamma = 0.8$) when the constraint does not exist. (c) Repeating (a) with $\alpha = 20$ to see clear convergence behavior of the meta-scheduler.

VII. Conclusion

In this paper we move from the traditional approach of designing a good downlink wireless scheduler given a scenario and/or rewards to that of determining which amongst a set of possible (good) schedulers is the best for the given context, e.g., user loads, service capacity, and performance requirements. Our, so called, meta-scheduler, provides a systematic approach to achieve robustness to uncertainty in the demand, environment or users' needs. This is accomplished by leveraging a budgeted multi-armed bandit framework, which uses the queuing system's regeneration cycles as natural times to make choices amongst arms (scheduling policies), but also by introducing a cycle interruption policy that is shown to ensure that eventually only stable policies are chosen. We provide a theoretical analysis which shows two objectives are met: (1) the approach has sub-linear regret, and (2) the losses due to interruptions are negligible. Our simulations show the meta-scheduler approach is effective, and exhibits the ability to achieve robust decisions in selecting a context-dependent best scheduling policy.

Finally, there has been a renewed interest in using Reinforcement Learning (RL) algorithms for wireless resource allocation. However, designing the ideal wireless scheduler that will achieve optimal performance in all possibly settings is likely an impossible goal, even with current RL techniques. Our meta-scheduler framework provides an approach to leverage a collection of state-of-art schedulers (possibly even RL based) which are known to be good for specific settings, and achieve "universality" by learning which amongst these provides the best results for the given operational scenario.

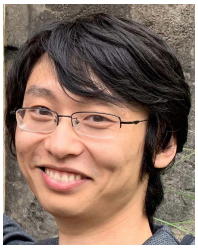
References

[1] J. Song, G. de Veciana, and S. Shakkottai, "Meta-scheduling for the wireless downlink through learning with bandit feedback," in *Proc. IEEE WIOPT Workshop Mach. Learn. Wireless Commun. (WMLC)*, Jun. 2020, pp. 1–7.

[2] R. Srikant and L. Ying, *Communication Networks: An Optimization, Control, and Stochastic Networks Perspective*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[3] P. Viswanath, D. N. C. Tse, and R. Laroia, "Opportunistic beamforming using dumb antennas," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1277–1294, Jun. 2002.

[4] X. Liu, E. K. P. Chong, and N. B. Shroff, "Opportunistic transmission scheduling with resource-sharing constraints in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 10, pp. 2053–2064, Oct. 2001.
[5] A. L. Stolyar, "On the asymptotic optimality of the gradient scheduling algorithm for multiuser throughput allocation," *Oper. Res.*, vol. 53, no. 1, pp. 12–25, Jan. 2005.
[6] I.-H. Hou and P. R. Kumar, "Packets with deadlines: A framework for real-time wireless networks," *Synth. Lectures Commun. Netw.*, vol. 6, no. 1, pp. 1–116, May 2013.
[7] I.-H. Hou, "Scheduling heterogeneous real-time traffic over fading wireless channels," *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1631–1644, Oct. 2014.
[8] M. Andrews *et al.*, "Scheduling in a queuing system with asynchronously varying service rates," *Probab. Eng. Inf. Sci.*, vol. 18, no. 2, pp. 191–217, 2004.
[9] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. Inf. Theory*, vol. 39, no. 2, pp. 466–478, Mar. 1993.
[10] S. Shakkottai and A. L. Stolyar, "Scheduling for multiple flows sharing a time-varying channel: The exponential rule," *Transl. Amer. Math. Soc.*, vol. 207, pp. 185–202, Dec. 2002.
[11] B. Sadiq, S. J. Baek, and G. de Veciana, "Delay-optimal opportunistic scheduling and approximations: The log rule," *IEEE/ACM Trans. Netw.*, vol. 19, no. 2, pp. 405–418, Apr. 2011.
[12] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. 15th ACM Workshop Hot Topics Netw.*, Nov. 2016, pp. 50–56.
[13] E. Cesar Santos, "A simple reinforcement learning mechanism for resource allocation in LTE networks with Markov decision process and Q-learning," 2017, *arXiv:1709.09312*. [Online]. Available: http://arxiv.org/abs/1709.09312
[14] T. Zhang, S. Shen, S. Mao, and G.-K. Chang, "Delay-aware cellular traffic scheduling with deep reinforcement learning," in *Proc. IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.
[15] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.
[16] R. Balakrishnan, K. Sankhe, V. S. Somayazulu, R. Vannithamby, and J. Sydir, "Deep reinforcement learning based traffic-and channel-aware OFDMA resource allocation," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.
[17] B. Liu, Q. Xie, and E. Modiano, "Reinforcement learning for optimal control of queueing systems," in *Proc. 57th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2019, pp. 663–670.
[18] D. Shah, Q. Xie, and Z. Xu, "Stable reinforcement learning with unbounded state space," 2020, *arXiv:2006.04353*. [Online]. Available: http://arxiv.org/abs/2006.04353
[19] B. Liu, Q. Xie, and E. Modiano, "RL-QN: A reinforcement learning framework for optimal control of queueing systems," 2020, *arXiv:2011.07401*. [Online]. Available: http://arxiv.org/abs/2011.07401
[20] L. Huang, X. Liu, and X. Hao, "The power of online learning in stochastic network optimization," in *Proc. ACM Int. Conf. Meas. Model. Comput. Syst.*, 2014, pp. 153–165.
[21] T. Chen, A. Mokhtari, X. Wang, A. Ribeiro, and G. B. Giannakis, "Stochastic averaging for constrained optimization with application to online resource allocation," *IEEE Trans. Signal Process.*, vol. 65, no. 12, pp. 3078–3093, Jun. 2017.
[22] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Found. Trends Mach. Learn.*, vol. 5, no. 1, pp. 1–122, 2012.
[23] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2020.
[24] A. Badanidiyuru, R. Kleinberg, and A. Slivkins, "Bandits with knapsacks," in *Proc. IEEE 54th Annu. Symp. Found. Comput. Sci.*, Oct. 2013, pp. 207–216.
[25] L. Tran-Thanh, A. Chapman, A. Rogers, and N. R. Jennings, "Knapsack based optimal policies for budget–limited multi–armed bandits," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 1134–1140.
[26] Y. Xia, H. Li, T. Qin, N. Yu, and T.-Y. Liu, "Thompson sampling for budgeted multi-armed bandits," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 3960–3966.
[27] S. Agrawal and N. Devanur, "Linear contextual bandits with knapsacks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3450–3458.
[28] S. Cayci, A. Eryilmaz, and R. Srikant, "Learning to control renewal processes with bandit feedback," *ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 2, p. 43, 2019.
[29] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, pp. 235–256, Dec. 2002.
[30] Y. Gai, B. Krishnamachari, and R. Jain, "Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation," in *Proc. IEEE Symp. New Frontiers Dyn. Spectr. (DySPAN)*, Apr. 2010, pp. 1–9.
[31] S. H. A. Ahmad and M. Liu, "Multi-channel opportunistic access: A case of restless bandits with multiple plays," in *Proc. 47th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2009, pp. 1361–1368.
[32] Y.-H. Kao, K. Wright, B. Krishnamachari, and F. Bai, "Online learning for wireless distributed computing," 2016, *arXiv:1611.02830*. [Online]. Available: http://arxiv.org/abs/1611.02830
[33] Y. Sun, X. Guo, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Learning-based task offloading for vehicular cloud computing systems," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–7.
[34] I. Tariq, R. Sen, G. D. Veciana, and S. Shakkottai, "Online channel-state clustering and multiuser capacity learning for wireless scheduling," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2019, pp. 136–144.
[35] B. Hajek, "Hitting-time and occupation-time bounds implied by drift analysis with applications," *Adv. Appl. Probab.*, vol. 14, no. 3, pp. 502–525, Sep. 1982.
[36] K. Liu and Q. Zhao, "Extended UCB policy for multi-armed bandit with light-tailed reward distributions," 2011, *arXiv:1112.1768*. [Online]. Available: http://arxiv.org/abs/1112.1768
[37] M. Series, "Guidelines for evaluation of radio interface technologies for IMT-advanced," ITU, Geneva, Switzerland, Tech. Rep. ITU 638, 2009.



Jianhan Song received the B.S. degree in information engineering from The Chinese University of Hong Kong in 2017. He is currently pursuing the joint M.S. and Ph.D. degree with the Department of Electrical and Computer Engineering, The University of Texas at Austin. He also joins the Wireless Networking and Communications Group (WNCG), The University of Texas at Austin, as a Graduate Research Assistant. His research interests include queuing systems, wireless networking, and reinforcement learning.

Gustavo de Veciana (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of California at Berkeley in 1987, 1990, and 1993, respectively. He served as the Director and the Associate Director for the Wireless Networking and Communications Group (WNCG), The University of Texas at Austin, from 2003 to 2007, where he is currently a Professor and the Associate Chair of the Department of Electrical and Computer Engineering. His research focuses on the design, analysis and control networks, information theory, and applied probability. His current interests include measurement, modeling and performance evaluation, wireless and sensor networks, architectures and algorithms to design reliable computing, and network systems. He was a recipient of the Cockrell Family Regents Chair in Engineering, the National Science Foundation CAREER Award 1996, and the IEEE Bennet Prize in 2021, and a co-recipient of six best paper awards. He has been an Editor for the IEEE/ACM TRANSACTIONS ON NETWORKING. He also serves on the board of trustees of IMDEA Networks Madrid.

Sanjay Shakkottai (Fellow, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, in 2002. He is currently with The University of Texas at Austin, where he is a Professor with the Department of Electrical and Computer Engineering, and holds the Cockrell Family Chair in Engineering #15. His research interests lie at the intersection of algorithms for resource allocation, statistical learning, and networks, with applications to wireless communication networks and online platforms. He received the NSF CAREER Award in 2004. He was a co-recipient of the IEEE Communications Society William R. Bennett Prize in 2021.