Opportunistic Spectrum Access: Does Maximizing Throughput Minimize File Transfer Time?

Jie Hu Vishwaraj Doshi Do Young Eun

***Abstract*—The *Opportunistic Spectrum Access* (OSA) model has been developed for the secondary users (SUs) to exploit the stochastic dynamics of licensed channels for file transfer in an opportunistic manner. Common approaches to design channel sensing strategies for throughput-oriented applications tend to maximize the long-term throughput, with the hope that it provides reduced file transfer time as well. In this paper, we show that this is not correct in general, especially for small files. Unlike prior delay-related works that seldom consider the heterogeneous channel rate and bursty incoming packets, our work explicitly considers minimizing the file transfer time of a *single* file consisting of multiple packets in a set of heterogeneous channels. We formulate a mathematical framework for the *static* policy, and extend to *dynamic* policy by mapping our file transfer problem to the stochastic shortest path problem. We analyze the performance of our proposed static optimal and dynamic optimal policies over the policy that maximizes long-term throughput. We then propose a heuristic policy that takes into account the performance-complexity tradeoff and an extension to online implementation with unknown channel parameters, and also present the regret bound for our online algorithm. We also present numerical simulations that reflect our analytical results.**

I. Introduction

In recent years, rapid growth in the number of wireless devices, including mobile devices and Internet of Things (IoT) devices has led to an explosion in the demand for wireless service. This demand further exacerbates the scarcity of allocated spectrum, which is ironically known to be underutilized by licensed users [1]. The *Opportunistic spectrum access* (OSA) model has been proposed to reuse the licensed spectrum in an opportunistic way otherwise wasted by licensed users [1]. Recently, the FCC has released a new guidance in 2020, which would expand the ability of the unlicensed devices (especially IoT devices) to operate in the TV-broadcast bands [2]. Besides, the related IEEE 802.22 family has been developed to enable spectrum sharing [3] to bring broadband access to rural areas.

In the OSA model, a *secondary user* (SU) aims to opportunistically access the spectrum when it is not used by any other users, while also prioritizing the needs of the *primary user* (PU). The SUs need to periodically sense the spectrum to avoid interfering with PUs. Interference reduces the quality of service in the OSA networks, and *throughput* is one of the most commonly used performance metrics in the OSA literature. The PU's behavior in a channel can be modeled as a Markov process (thus correlated over time), for which partially observable Markov decision processes (POMDPs) are typically employed to formulate the spectrum sensing strategy in order to maximize the long-term throughput [4]. These POMDPs do not possess known structured solutions in general and they are known to be PSPACE-complete even if all the channel statistics are known a priori [5]. To achieve near maximum throughput, computationally efficient yet sub-optimal policies, such as myopic policy [6] and Whittle's index policy [7], have been proposed for the *offline* OSA setting (known channel parameters) and later extended to the *online* setting (unknown channel parameters) [8]. Besides, single-channel online policies have also been developed in [9], [10] to find the channel with maximum throughput in the steady state. Recently, by focusing more on heterogeneous channels with *i.i.d* Bernoulli distribution for each, multi-armed bandit (MAB) techniques have been extensively studied to let SUs learn unknown channel parameters on the fly, and have been known for their *lightweight* designs. MAB techniques that maximize the cumulative reward (or minimize the regret) can directly be applied to the online setting for maximizing the throughput, including the Bayesian approach [11], upper confidence bounds [12], thompson sampling [13] and its improvement from efficient sampling [14], and coordination approach among multiple SUs [15].

Nowadays, low latency has become one of the main goals for 5G wireless networks [16] and other time-sensitive applications with guaranteed delay constraints. For example, packet delay in cognitive radio networks has been extensively studied using queuing theory to derive delay-efficient spectrum scheduling strategies. In this setting, a stream of packet arrivals modeled as a Poisson process with a constant rate is a common assumption in delay related works [17]–[21], and the goal is often to minimize the average packet delay in the steady state. In reality, however, so-called 'bursty' cases are also commonly seen, where a finite number of packets comprising one file can be pushed into the SU's queue simultaneously and transmitted opportunistically by the SU, and the next 'file' will not arrive if the current file transfer job has not been finished yet.[1] For instance, the IEEE 802.11p MAC protocol requires the safety message to be generated and sent by each vehicle in every 100 ms interval [22], Poisson arrivals being unsuitable to model such situation. Same channel rate across all channels is another implicit assumption in many works [18]–[21], but it

Jie Hu and Do Young Eun are with the Department of Electrical and Computer Engineering, and Vishwaraj Doshi is with the Operations Research Graduate Program, North Carolina State University, Raleigh, NC. Email: {jhu29, vdoshi, dyeun}@ncsu.edu. This work was supported in part by National Science Foundation under Grant CNS-1824518 and IIS-1910749.

[1] One packet transmission in the delay-related studies [17]–[21] is equivalent to a single file transfer job in the 'bursty' case because a single file in the SU's queue can be seen as a 'large' packet.

doesn't reflect the realistic heterogeneous channel environment frequently assumed in the throughput-oriented studies in the OSA literature [10]–[14]. Clearly, allowing the SU to switch across heterogeneous-rate channels during instances of PU's interruption can further reduce the file transfer time, but to the best of our knowledge, this issue has not been fully explored.[2]

In this paper, we study the OSA model with the aim of minimizing the transfer time of a single file over heterogeneous channels. The common folklore assumes that the policy maximizing the long-term throughput would also lead to the minimum expected file transfer time. For example, Wald's equation implies that the file download time in an *i.i.d* (over time) channel is equal to the file size divided by the average throughput of that channel, implicitly favoring the max-throughput channel for minimal download time. In this paper, we show that this is *not* the case in general, even in the *i.i.d* (over time) channel. We first present a theoretical analysis for *static* policies where the SU only sticks to one channel throughout the entire file transfer. By casting the file transfer problem into a stochastic shortest path framework, the SU is free to switch between channels and we are able to obtain the dynamic *optimal* policy. Our theoretical analysis shows that static and dynamic optimal policies reduce the transfer time compared to the baseline max-throughput policy, and this reduction is even more significant in delay-sensitive applications, where files are relatively small. We then propose a lightweight heuristic policy with good performance and extend to an online implementation with unknown channel parameters the SU needs to learn on the fly. We modify a MAB algorithm (model-based method) proposed in [23] in the online setting and show its gap-dependent regret bound, instead of the model-free reinforcement learning used in delay-related works [18], [20] for sample efficiency purpose [24]. We also use simulation results to visualize that the max-throughput policy is not the best when it comes to achieving the minimum file transfer time.

The rest of the paper is organized as follows: In Section II we introduce the OSA model and characterize the file transfer problem and its policy under the OSA framework. In Section III, we show the expected transfer time for static policy and it's performance analysis. Then, we extend from the static policy to the dynamic policy in Section IV. The practical concerns are discussed in Section V. Finally in Section VI, we evaluate different policies in the realistic numerical setting.

II. Model description

A. The OSA Model

Consider a set of N heterogeneous channels $\mathcal{N} \triangleq \{1, 2, \cdots, N\}$ available for use and each channel $i \in \mathcal{N}$ offers a stable rate of $r_i > 0$ bits/s if successfully utilized [7], [10]. In our setting, a SU wishes to transfer a file of size F bits using one of these N channels via opportunistic spectrum access. The SU can only access one channel at any given time, and can maintain this access for a fixed duration of Δ seconds, after which it has to sense available channels again in order to access them. At this point, the SU can decide which channel to sense and access that channel for the next Δ second interval if the channel is *available*. Or the SU has to wait for Δ seconds to sense again if that channel is *unavailable*, thereby unable to transfer data for this duration. This pattern is known as the *constant access time* model, and has been commonly adopted for the SU's behavior as a collision prevention mechanism in the OSA literature [1], [11]. The cycle repeats itself until the SU transmits the entire file size F, then it immediately exits the channel in use. Note that the duration Δ seconds is not a randomly chosen number. For example, Δ is recommended as 100 ms because the SU needs to vacate the current channel within 100 ms once the PU shows up, as defined in IEEE 802.22 standard [3]. The SU can transmit up to 3.1 Mb in each Δ seconds with highest channel rate 31 Mbps in IEEE 802.22 standard and many small files (e.g, 5 KB text-only email, 800 KB GIF image) need just a few slots to complete.

We say a channel is unavailable (or *busy*) if it is currently in use by the primary users (PUs) or other SUs, while it is available (or *idle*) if it is not in use by any other users. The state of a channel (idle or busy) is assumed to be independent over all channels $i \in \mathcal{N}$, and *i.i.d.* over the time instants $\{0, \Delta, 2\Delta, \cdots\}$ following Bernoulli distribution with parameter $p_i \in (0, 1]$, in line with the widely used discrete-time channel model [1]. Specifically, for each $i \in \mathcal{N}$, $\{Y_i(k)\}_{k\in\mathbb{N}}$ is a Bernoulli process with $p_i = P[Y_i(k) = 1] = 1 - P[Y_i(k) = 0]$ for all $k \in \mathbb{N}$. Then, we can define $X_i(t)$, the state of channel $i \in \mathcal{N}$ at any time $t \in \mathbb{R}_+$, as a piecewise constant random process: $X_i(t) \triangleq Y_i(\lfloor t/\Delta \rfloor)$, where $\lfloor \cdot \rfloor$ denotes the floor function. This way, we write $X_i(t) = 1$ (0) if channel $i \in \mathcal{N}$ is available (unavailable) for the SU with probability p_i $(1 - p_i)$.

In this setup, the rate at which the SU can transmit files through channel $i \in \mathcal{N}$ at any time instant $t \geq 0$, also termed as the *instantaneous throughput* of the channel i, is given by $r_i X_i(t)$, with its *throughput* [12], [14] by $\mathbb{E}[r_i X_i(t)] = r_i p_i$. We denote by $i^* \triangleq \arg\max_{k\in\mathcal{N}} r_k p_k$ the channel with the *maximum throughput*. For simplicity, we assume that this channel is unique, i.e., $r_{i^*} p_{i^*} > r_k p_k$ for all $k \in \mathcal{N}\backslash\{i^*\}$. In what follows, we introduce some basic notations and expressions regarding *policies* under this OSA framework.

B. Policies for File Transfer

We define a *policy* at time t to be a mapping $\pi : \mathbb{R}_+ \mapsto \mathcal{N}$ where $\pi(t) = i$ indicates that the SU has chosen channel i to access during the time period $\big[\lfloor \frac{t}{\Delta} \rfloor \Delta, \big(\lfloor \frac{t}{\Delta} \rfloor + 1\big)\Delta\big)$. From our standing assumption, a policy therefore only changes at $t \in \{0, \Delta, 2\Delta, \cdots\}$, and all policies ensure that the file transfer for any finite size F will eventually be completed. This way, the policy $\pi(t)$ is a piecewise constant function (mapping), defined at all time $t \geq 0$. For a given policy π, let $T(\pi, F)$ denote the *transfer time* of a file of size F — the entire duration of time to complete the file transfer, which is written as $T(\pi, F) = \min_{T\geq 0}\{\int_0^T r_{\pi(t)} X_{\pi(t)}(t)dt \geq F\}$. Figure 1 explains the file transfer progress via OSA model.

The objective of our OSA framework is to minimize the expected transfer time $\mathbb{E}[T(\pi, F)]$ over the set of all policies π.

[2]Although [17] assumes heterogeneous channel rates, it requires the SU to stick to one channel to complete the packet transmission, even if the transmission may be interrupted by the PU multiple times.

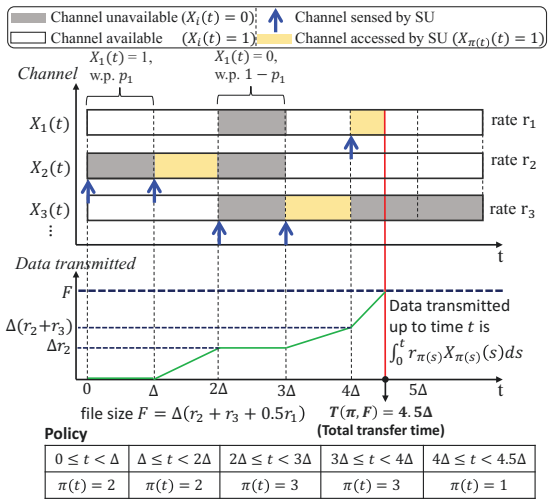


Fig. 1. File transfer via the OSA framework. The SU senses channels according to its policy, and accesses the channel if it is available. Upon gaining access, it begins transmitting data at the corresponding channel rate. Transmission ends (red line) as soon as the amount of data transmitted (green line) equals the file size F.

Policies can be *static*, where the SU only senses and transmits via one (pre-determined) channel i throughout the file transfer, that is, $\pi(t)=i$ for all $t\geq 0$. For such static policies, we denote by $T(i,F)$ their transfer time for file size F. The channel that provides the minimum expected transfer time is then called *static optimal* given by

$$i_{so}(F)=\arg\min_{k\in\mathcal{N}}\mathbb{E}[T(k,F)],\quad \text{(static optimal)} \qquad (1)$$

and we denote $T(i_{so},F)$ the corresponding transfer time for this static optimal policy. Note that the static optimal channel $i_{so}(F)$ depends on the file size F and can vary for different file sizes. Policies can also be *dynamic*, in which an SU is allowed to change the channels it chooses to sense throughout the course of the file transfer. Given a file size F, the policy with the minimum expected transfer time over the set of all policies $\Pi(F)$ is called the *dynamic optimal* policy given by

$$\pi^*(F)=\arg\min_{\pi\in\Pi(F)}\mathbb{E}[T(\pi,F)]. \text{ (dynamic optimal)} \qquad (2)$$

Lastly, we define the *max-throughput policy* as the static policy with the channel i^*, which maximizes the long-term throughput. In the next section, we take a closer look at the max-throughput policy and static policies in general.

III. Static Optimal Policy

Recent works in the OSA literature focus on estimating channel parameters p_i's, with the goal of eventually converging to the policy $i^*=\arg\max_{k\in\mathcal{N}} r_k p_k$ which provides the maximum throughput [10]–[14].[3] They focus on minimizing the 'regret' in the MAB model, defined as the difference between the cumulative reward obtained by the online algorithm and the max-throughput policy (the optimal policy in hindsight).

[3]While [12]–[14] deal with link rate selection problem to select best rate in one channel to maximize the expected throughput, the mathematical model of link rate selection problem is essentially the same as the standard OSA setting for choosing the max-throughput channel, as considered in our setting.

The essential assumption behind all these approaches is that the SU always fully dedicates Δ seconds in each time interval for file transfer. Channel i^* appears as a good candidate since it provides the largest expected data transfer $\Delta r_{i^*}p_{i^*}$ across every time interval. This is further supported by the well-known Wald's equation with the *i.i.d* reward assumption at each time interval, suggesting that $\mathbb{E}[T(i,F)]=F/r_ip_i$ for each channel $i\in\mathcal{N}$, which is then minimized by i^*. When policies are dynamic, however, the rewards are not identically distributed since the transfer rates of the dynamically accessed channels can be different, making Wald's equation inapplicable. Surprisingly, it is not applicable for static policies either. As typically is the case in delay-sensitive applications [17]–[19], [21], the file sizes are often not that large, rendering their transfer times small enough that an SU may not need to utilize the whole Δ seconds for data transfer in each time interval. The reward summands are still not identically distributed, causing Wald's equation to fail in general.

Our key observation in this paper is that choosing channel i^* may not be the best option to minimize the expected transfer time. In this section, we limit ourselves to the set of static policies of the form shown in (1) and analyze the resulting expected transfer time in the OSA network. We use this to compare the performance gap between the max-throughput policy and the static optimal policy, and show that for a reasonable choice of channel statistics and file sizes, the static optimal policy performs significantly better than the max-throughput policy. We derive a closed-form expression of the expected transfer time of a file of size F in each fixed channel by the following proposition.

Proposition 3.1: Given a file of size F, the expected transfer time $\mathbb{E}[T(i,F)]$ of the static policy for channel $i\in\mathcal{N}$ is

$$\mathbb{E}[T(i,F)]=\Delta\left(k_i/p_i+\mathbb{1}_{\{\alpha_i>0\}}(1-p_i)/p_i+\alpha_i\right), \qquad (3)$$

where $k_i\triangleq\lfloor F/\Delta r_i\rfloor\in\mathbb{Z}_+$ and $\alpha_i\triangleq F/\Delta r_i-k_i\in[0,1)$.

Sketch of proof: Since each channel is *i.i.d* Bernoulli distributed, the waiting time until one access to any channel $i\in\mathcal{N}$ is a geometrically distributed random variable, independent across all channels, with mean $\Delta(1-p_i)/p_i$. Each file size F transmitted in channel i needs to access channel i for k_i times. If the remaining portion α_i is nonzero, then the SU needs additional random waiting time to complete the transfer. Combining the total transmission time F/r_i and the expected total waiting time gives (3). ■

We have from Proposition 3.1 that $\mathbb{E}[T(i,F)]$, the expected transfer time for any file size F under the static policy on channel $i\in\mathcal{N}$, can be explicitly written in terms of file size F, time duration Δ and channel statistics r_i and p_i of the chosen channel i. Substituting $k_i=F/\Delta r_i-\alpha_i$ in (3) gives

$$\mathbb{E}[T(i,F)]=\frac{F}{r_ip_i}+\Delta\mathbb{1}_{\{\alpha_i>0\}}(1-\alpha_i)\frac{1-p_i}{p_i}\geq\frac{F}{r_ip_i}. \qquad (4)$$

The inequality in (4) shows the expected transfer time of any static policy is no smaller than that given by Wald's equation.

We use Figure 2 to illustrate the results in Proposition 3.1, where each line represents the expected transfer time via one channel over a range of file sizes from (3). We observe that the

expected transfer time of channel 1 (red line) is always above the Wald's equation of channel 1 (purple dot-line). As shown in (3), the slope of each line (channel i) is $1/r_i$ and the 'jump size' $\Delta(1-p_i)/p_i$ is equal to the expected waiting time till the channel is available. The jumps in the plot for each channel i, representing the waiting times, occur at exactly the instances where file size is an integer multiple of Δr_i, and come into play especially when there is still a small amount of remaining file to be transferred at the end of a Δ time interval.

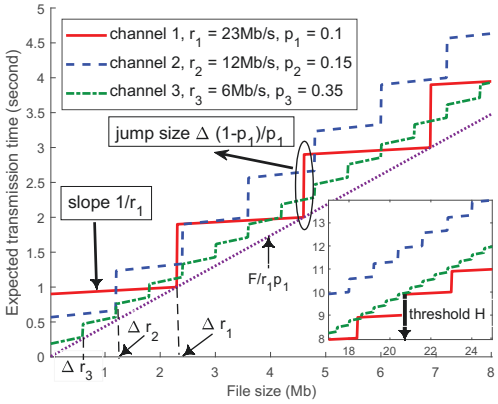


Fig. 2. Expected transfer time from (3) with duration $\Delta = 100$ ms (IEEE 802.22 standard). Channel 1 has the maximum throughput. The purple dot-line $E[T] = F/r_1p_1$ corresponds to the Wald's equation for channel 1. Downward arrow in the inset figure is the threshold H in Proposition 3.2.

By definition, the static optimal policy provides the minimum expected transfer time over all static policies including the max-throughput policy itself. While it is true for all file sizes, in some cases with certain file sizes, these two policies may coincide.

Proposition 3.2: The max-throughput policy coincides with the static optimal policy, that is, $i_{so}(F) = i^*$, for any file size F satisfying at least one of the two conditions below:

1) F exceeds a threshold H, where
$$H = \frac{\Delta(1-p_{i^*})/p_{i^*}}{1/r_hp_h - 1/r_{i^*}p_{i^*}}, \tag{5}$$
and $h = \arg\max_{j\in\mathcal{N}\setminus\{i^*\}} r_jp_j$ is the channel with the second largest throughput.
2) F is an integer multiple of Δr_i^*, i.e., $F = k\Delta r_{i^*}$ for some $k \in \mathbb{Z}_+$.

Sketch of proof: From (4) we have the lower bound of $\mathbb{E}[T(i,F)]$, and the upper bound is derived from (3) with $\mathbb{1}_{\{\alpha_i>0\}}(1-\alpha_i) \le 1$. The sufficient condition to ensure $\mathbb{E}[T(i^*,F)] \le \mathbb{E}[T(i,F)]$ is that the upper bound of $\mathbb{E}[T(i^*,F)]$ is smaller than the lower bound of $\mathbb{E}[T(i,F)]$ for any channel i, which leads to (5). When $F = k\Delta r_{i^*}$, from (3) we have the expected transfer time $\mathbb{E}[T(i^*,F)] = F/r_{i^*}p_{i^*}$. Since $r_{i^*}p_{i^*} \ge r_jp_j$ for any $j \in \mathcal{N}$, we can show $\mathbb{E}[T(i^*,F)]$ is smaller than the lower bound of $\mathbb{E}[T(i,F)]$ for any channel i, which establishes condition 2), completing the proof. ∎

Outside of Proposition 3.2, however, there are many instances where the max-throughput channel is not static optimal and other channels can perform better for smaller file sizes. In such cases, we would like to discuss how much time the static optimal policy can save against the max-throughput policy.

Corollary 3.3: Let $m_i \triangleq p_i/p_{i^*}$ for $i \in \mathcal{N}\setminus\{i^*\}$. Consider a file of size $F \in (k\Delta r_{i^*}, (k+1)\Delta r_{i^*})$ for some $k \in \mathbb{N}$. Then, we have
$$\frac{\mathbb{E}[T(i_{so},F)]}{\mathbb{E}[T(i^*,F)]} \le \min_{i\in\mathcal{N}\setminus\{i^*\}} \left\{1, \frac{F/\Delta r_ip_i + (1-p_i)/p_i}{(k+1)(m_i/p_i - 1)}\right\}. \tag{6}$$

Note that by definition $m_i/p_i = 1/p_{i^*} > 1$ so that the upper bound on the ratio $\mathbb{E}[T(i_{so},F)]/\mathbb{E}[T(i^*,F)]$ in (6) is always in the interval $(0,1]$. Moreover, smaller ratio means better performance of the static optimal policy against the max-throughput policy. To gauge how the parameters of the max-throughput channel could affect the performance of the static optimal policy, suppose we fix F, Δ, r_i, p_i for all $i \in \mathcal{N}\setminus\{i^*\}$ and the maximum throughput $r_{i^*}p_{i^*}$, while treating p_{i^*} as a variable. The upper bound in (6) is then monotonically decreasing in $m_i = p_i/p_{i^*}$, and can even approach 0 if at least one of m_i is really large, resulting in the huge performance gain of the static optimal channel compared to that of the max-throughput channel. This implies that accessing channel i^* can take much longer time to transmit a file than other channels if its available probability p_{i^*} is very small, which is common in outdoor networks where the max-throughput channel i^* has very high rate but with low available probability [25].

Our static optimal policy shows better performance against the max-throughput policy for small files and small p_{i^*}. Since the static optimal channel depends on the file size, choosing channels *dynamically* according to its remaining file size can further reduce the expected transfer time. We next formulate the file transfer problem as an instance of the stochastic shortest path (SSP) problem and analyze the performance of the dynamic optimal policy.

IV. Dynamic Optimal Policy

Now that we have analyzed the static policies, we turn our attention to feasible dynamic policies for our file transfer problem. We start by first formulating the file transfer problem as a stochastic shortest path (SSP) problem, in which the agent acts dynamically according to the stochastic environment to reach the predefined destination as soon as possible. Then, we translate this SSP problem into an equivalent shortest path problem, which helps us derive the closed-form expression of the expected transfer time for any given dynamic policy, and we utilize this to obtain the performance analysis of the dynamic optimal policy against the max-throughput policy.

A. Stochastic Shortest Path Formulation

The SSP problem is a special case of the infinite horizon Markov decision process [26]. To make this section self-contained, we explain our problem as a SSP problem.

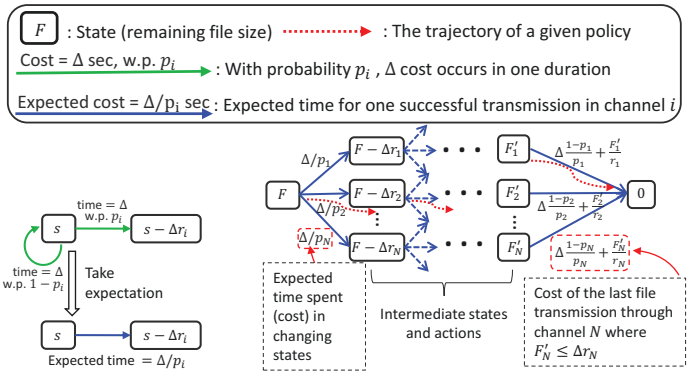
State Space and *Action Space:* We define the state $s \in \mathcal{S} \triangleq \mathbb{R}_+$ of our SSP as the remaining file size yet to be transmitted. The action $i \in \mathcal{N}$ is the channel chosen to be sensed at the beginning of each time interval. The objective of our problem is to take the optimal action at each state s which minimizes the expected time to transmit the file of size F.

State Transition: Denote by $P_{s,s'}(i)$ the transition probability that the SU moves to state s' after taking action i at state s.

From any given state $s \in \mathcal{S}\setminus\{0\}$, the next state under any action $i \in \mathcal{N}$ depends on the availability of channel i. Since the channel is available or unavailable according to an *i.i.d* (over time) Bernoulli distribution, the next state is either the same as the current one if channel i is unavailable, i.e. $P_{s,s}(i) = 1-p_i$; or the next state is $(s-\Delta r_i)^+ \triangleq \max\{0, s-\Delta r_i\}$ if channel i is available, i.e. $P_{s,(s-\Delta r_i)^+}(i) = p_i$. State 0 is a termination state since there is no file transmission remaining.

Cost Function: The cost $c(s,i,s')$ is the amount of time spent in transition from state s to s' after sensing channel i. Since the SU can only sense channels at intervals of size Δ, sensing an unavailable channel costs a Δ second waiting period until the SU can sense next, that is, $c(s,i,s) = \Delta$ for all $s \in \mathcal{S}\setminus\{0\}, i \in \mathcal{N}$. Similarly, if the sensed channel is available, the time spent in transmitting is also Δ seconds, unless the SU finishes transmitting the file early. In the latter case the cost of transmission is $c(s,i,0) = s/r_i$. Overall, the cost of a successful transmission can be written as $c(s,i,(s-\Delta r_i)^+) = \min\{\Delta, s/r_i\}$ for all $s \in \mathcal{S}\setminus\{0\}, i \in \mathcal{N}$. Once the remaining file size reduces to 0, the SU will end this file transmission immediately with no additional cost incurred, so that $c(0,i,0) = 0$ for any $i \in \mathcal{N}$.

Our dynamic policy[4] is written as a mapping $\pi : S \to \mathcal{N}$, where $\pi(s) \in \mathcal{N}$ denotes the channel chosen for sensing when the current state (remaining file size) is s. For any policy π, we have $T(\pi,0) = 0$ at the termination state. Our goal in this SSP problem is to find the dynamic optimal policy $\pi^*(F)$ that minimizes the expected transfer time for the file size F, which can be derived from a variety of methods such as value iteration, policy iteration and dynamic programming [26].



(a) Transformation of the SSP problem.

(b) Equivalent shortest path diagram.

Fig. 3. Illustration to translate the file transfer problem into an equivalent shortest path problem.

B. Performance Analysis

For ease of exposition, we introduce additional notations here. By a *successful transmission*, we refer to state transitions of the form $s \to s'$. This is denoted by the horizontal green line in Figure 3(a) connecting states s and $s' \triangleq s - \Delta r_i > 0$, and should be distinguished from the self-loop $s \to s$, which implies the sensed channel was unavailable. As shown in Figure 3(a), taking expectation helps get rid of these self-loops by casting the original SSP to a deterministic shortest path problem in expectation. The cost associated with each link is then the expected time it takes to transit between the states. Figure 3(b) shows the underlying network for the shortest path problem, where each link is a channel chosen to be sensed and each path from *source* F to *destination* 0 corresponds to a policy $\pi \in \Pi(F)$. The path-length or the number of links traversed from F to 0 under any given policy π then becomes the total number of successful transmissions needed by that policy to complete the file transfer, which we denote by $|\pi|$.

For any policy $\pi \in \Pi(F)$ and $n \in \{1, \cdots, |\pi|\}$, let F_n denote the remaining file size right before the n-th successful transmission. Then for all $n \in \{2, \cdots, |\pi|\}$, we have the recursive relationship: $F_n = F_{n-1} - \Delta r_{\pi(F_{n-1})}$, starting with $F_1 = F$ and ending with $F_{|\pi|+1} = 0$. Given a file size F, each policy $\pi \in \Pi(F)$ can then be written in a vector form as $\boldsymbol{\pi} = [\pi(F_1), \pi(F_2), \ldots \pi(F_{|\pi|})]^T$. With this in mind, we can derive a closed-form expression of the expected transfer time for any dynamic policy in the following proposition.

Proposition 4.1: Given a file of size F, the expected transfer time of a dynamic policy π is written as

$$\mathbb{E}[T(\pi,F)] = \Delta\left(\sum_{n=1}^{|\pi|-1} \frac{1}{p_{\pi(F_n)}}\right) + \Delta\frac{1-p_{\pi(F_{|\pi|})}}{p_{\pi(F_{|\pi|})}} + \frac{F_{|\pi|}}{r_{\pi(F_{|\pi|})}}. \quad (7)$$

In (7), the first summation is the cumulative expected transmission time, or the cost, to the $(|\pi|-1)$-th successful transmission, with the last two terms being the expected transmission time of the last successful transmission. Proposition 4.1 also includes the expected transfer time of the static policy as a special case. Recall that $k_i = \lfloor F/\Delta r_i \rfloor$ and $\alpha_i = F/\Delta r_i - k_i$ in Proposition 3.1. When applied to a static policy for any channel $i \in \mathcal{N}$, we have $|\pi| = k_i + \mathbb{1}_{\{\alpha_i > 0\}}$ and $p_{\pi(F_n)} = p_i$ for all $n = 1, 2, \cdots, |\pi|$. The recursive relationship becomes: $F_n = F_{n-1} - \Delta r_i$, implying that $F_n = F - (n-1)\Delta r_i$. Then, we have $F_{|\pi|} = F - (|\pi|-1)\Delta r_i = \alpha_i \Delta r_i$ if $\alpha_i > 0$. Otherwise, $F_{|\pi|} = \Delta r_i$. Substituting these into (7) gets us (3).

The common folklore around the max-throughput policy is that it would lead to the minimum expected file transfer time of $F/r_{i^*}p_{i^*}$. Our next result shows this is too optimistic and not achieved in general even under the dynamic optimal policy.

Proposition 4.2: For any file size F and any dynamic policy $\pi \in \Pi(F)$, we have $\mathbb{E}[T(\pi,F)] \ge \mathbb{E}[T(\pi^*,F)] \ge F/r_{i^*}p_{i^*}$. Moreover, $i^* = \pi^*(F)$ for $F = k\Delta r_{i^*}, k \in \mathbb{Z}_+$.

Sketch of proof: From the definition of $F_{|\pi|}$ we have $F_{|\pi|} \le \Delta r_{\pi_{|\pi|}}$. Using $r_{i^*}p_{i^*} \ge r_i p_i$ for all $i \in \mathcal{N}$ and $F_{|\pi|} \le \Delta r_{\pi_{|\pi|}}$, together with (7), gives $\mathbb{E}[T(\pi,F)] \ge \mathbb{E}[T(\pi^*,F)] \ge F/r_{i^*}p_{i^*}$. When $F = k\Delta r_{i^*}, k \in \mathbb{Z}_+$, from (3.1) we also have $\mathbb{E}[i^*,F] = F/r_{i^*}p_{i^*}$ and $\mathbb{E}[i^*,F] \ge \mathbb{E}[T(\pi^*,F)]$ by definition. Using the squeeze theorem gives $i^* = \pi^*(F)$. ∎

As shown in (4), $F/r_{i^*}p_{i^*}$ is always the lower bound on the transfer time for any static policy. Proposition 4.2 strengthens this by showing that the same is true even for the dynamic optimal policy. Similar to condition (b) in Proposition 3.2 for the static optimal policy, the dynamic optimal policy $\pi^*(F)$ also coincides with the max-throughput policy i^* when the file

[4]There always exists an optimal policy π^* to be deterministic in the SSP problem, as proved in Proposition 4.2.4 [26]. Thus, we restrict ourselves to the class of deterministic policies in this paper.

size is an integer multiple of Δr_{i^*}, while we no longer have the finite threshold H as in Proposition 3.2(a). We next give bounds to quantify the performance of the dynamic optimal policy with respect to the max-throughput policy.

Corollary 4.3: Let $m_i \triangleq p_i/p_{i^*}$ for $i \in \mathcal{N} \setminus \{i^*\}$. Consider a file of size $F \in (k\Delta r_{i^*}, (k+1)\Delta r_{i^*})$ for some $k \in \mathbb{N}$. Then, we have

$$1/(1+\Delta \mathbb{1}_{\{\alpha_i>0\}}(1-p_{i^*})r_{i^*}/F) \leq \frac{\mathbb{E}[T(\pi^*,F)]}{\mathbb{E}[T(i^*,F)]}$$
$$\leq \min_{i\in\mathcal{N}\setminus\{i^*\}}\left\{1, \frac{F/\Delta r_i p_i + (1-p_i)/p_i - k m_i (r_{i^*}p_{i^*} - r_i p_i)/r_i p_i^2}{(k+1)(m_i/p_i - 1)}\right\}.$$

To better understand Corollary 4.3 we analyze how the parameters of the max-throughput channel could impact the performance of the dynamic optimal policy. Similar to Corollary 3.3, small value of $\mathbb{E}[T(\pi^*,F)]/\mathbb{E}[T(i^*,F)]$ implies that the dynamic optimal policy offers significant saving in time over the max-throughput policy. We note that Corollary 4.3 tightens the upper bound with an extra negative term in the numerator, compared to that in Corollary 3.3, potentially providing greater savings in time as we extend the policy from static optimal to the dynamic optimal.

In contrast to Proposition 3.2 that max-throughput policy is good enough for $F \geq H$, Corollary 4.3 tells us that there is always some reduction in file transfer time even for large file size F under the dynamic optimal policy. This is because the extra negative term in the numerator can be large, since $k = \lfloor F/\Delta r_{i^*} \rfloor$ could be big for large F, implying that the second argument in the $\min\{\cdot,\cdot\}$ function may no longer be increasing in F. Note however that the reduction in transfer time would be minimal for large file sizes since the lower bound in Corollary 4.3 will rise to 1 as F goes to infinity.

V. Practical considerations

While the dynamic optimal policy gives a smaller expected transfer time, we face a scaling problem when the file size can differ from each, effectively changing the underlying 'graph' in the corresponding shortest path problem. This warrants re-computation of the dynamic optimal policy for each file size, which would be unacceptable in reality. In this section, we discuss the policy re-usability issue and propose a heuristic policy balancing the performance and the computational cost. We also consider the case where the SU has no information about the channel parameters beforehand and it must sense and access channels *on the fly* in order to find the optimal policy, thereby extending the problem to an *online* setting.

A. Performance-Complexity Trade Off

Due to the nature of the shortest path problem, a change in file size induces a change in the underlying graph. Transmitting different-sized files is very common [27], and if the goal is to always determine the best solution, the only option is to recompute the dynamic optimal policy for every different file size. This would not be scalable in applications where minimal computation is required, and policies that can be promptly modified and reused across different file sizes with performance guarantees would be highly desirable.

We approach this issue by proposing a heuristic policy that utilizes the max-throughput policy, which is fixed and known to the SU, and the static optimal policy, which is relatively simpler to evaluate due to $E[T(i,F)]$ being known in closed form for any channel i and file size F. Note that the max-throughput policy coincides with the dynamic optimal policy when the file size is an integer multiple of Δr_{i^*} according to Proposition 4.2. We also know that the static optimal policy significantly outperforms the max-throughput policy especially for smaller file sizes. Combining these two policies, by transmitting file through max-throughput channel until the remaining file size becomes 'small' so as to apply the static optimal policy for the rest, will strike the right balance between computational complexity and achievable performance gain.

With this motivation in mind, we divide file size $F := F_1 + F_2$ into two parts: $F_1 = m\Delta r_{i^*}$ $(m = 0, 1, \cdots, \lfloor F/\Delta r_{i^*} \rfloor)$ and $F_2 = F - F_1$. The heuristic policy π_{heu} is defined as follows: The SU first transmits the F_1 amount of the file through the max-throughput channel i^*, and then sticks to the static optimal policy $i_{so}(F_2)$ for the remaining amount F_2 of the file. Due to the page limit, we refer readers to the proof of Corollary 4.3 in our technical report [28], in which the upper bound of ratio $E[T(\pi_{heu},F)]/E[T(i^*,F)]$ is monotonically decreasing in m. The largest possible m gives us the smallest upper bound of the ratio. Moreover, the upper bound in Corollary 4.3 is smaller than that of the static optimal policy in Corollary 3.3. These arguments suggest that the heuristic policy π_{heu} with largest possible m can potentially lead to smaller transfer time compared to different values of m. Besides, our heuristic policy can further reduce the computational cost for a set of files sharing the same remaining file size F_2 because $i_{so}(F_2)$ has already been found and no further re-computation is needed for this set of files.

B. Unknown Channel Environment

We now consider the setting where the SU does not know the available probability p_i for any channel $i \in \mathcal{N}$ and only knows the rate r_i — a commonly analysed setting in the OSA literature [10], [11]. The SU has no alternative but to observe the states of these channels when it tries to access them, and build its own estimations of channel probabilities. In this extended setting, we study our problem as an online shortest path problem, which has been widely studied in [23], [29], [30] for different kinds of cost functions. [23] proposed a Kullback-Leibler source routing (*KL-SR*) algorithm to an online routing problem with geometrically distributed delay in each link, which coincides with our link cost in the underlying graph of the shortest path problem shown in Figure 3(b).

For our purpose, we modify *KL-SR* algorithm; key differences being that we let the file size vary across the episodes, allowing a different underlying graph of the shortest path problem for each episode instead of the fixed underlying graph of the shortest path problem in [23]. Algorithm 1 describes our online implementation, where F^k is the file size to be transferred in the k-th episode. $n_i(k)$ denotes the number of times channel i has been sensed before the k-th episode and $\bar{p}_i(k)$ is the empirical average of channel i's available probability throughout the $k-1$

episodes so far. With $n_i(k)$ and $\bar{p}_i(k)$, the estimated available probability $\hat{p}_i(k)$ of channel i is then derived from the KL-based index in [23]. As mentioned in line 1 in Algorithm 1, the SU can choose one of the various policies according to which it wishes to perform the file transfer, i.e., dynamic optimal policy π^*, static optimal policy i_{so}, max throughput policy i^* or the heuristic policy π_{heu}, and then stick to that policy. Let $\tilde{\mathbb{E}}[T(\pi, F^k)]$ be the estimated expected transfer time of policy π at the k-th file by using the estimated parameter $\hat{p}_i(k)$ instead of p_i for all $i \in \mathcal{N}$ in (7). In line 7 in Algorithm 1, π^k will be computed as $\pi^k = \arg\min_{\pi \in \Pi(F^k)} \tilde{\mathbb{E}}[T(\pi, F^k)]$ for dynamic optimal policy; $\pi^k = \arg\min_{i \in \mathcal{N}} \tilde{\mathbb{E}}[T(i, F^k)]$ for static optimal policy and $\pi^k = \arg\max r_i \hat{p}_i(k)$ for max-throughput policy. For heuristic policy π_{heu}, π^k will be computed in the same way as described in Section V-A with estimated parameters $\{\hat{p}_i(k)\}_{i \in \mathcal{N}}$.

Algorithm 1 Online file transfer algorithm

1: Choose the type of policy to use: π^*, i_{so}, i^* or π_{heu}.
2: Apply static policy i in the i-th episode to transmit the file of size F^i for $i = 1, 2, \cdots, N$, and update $n_i(N+1)$, $\bar{p}_i(N+1)$ for $i \in \mathcal{N}$ at the end of the N-th episode
3: **for** $k \geq N+1$ **do**
4: Compute the estimated channel statistics $\{\hat{p}_i(k)\}_{i \in \mathcal{N}}$ according to the KL-based index in [23].
5: Given a file of size F^k, compute the policy π^k with $\{\hat{p}_i(k)\}_{i \in \mathcal{N}}$ and observe channel status for the whole file transfer process in this episode.
6: Update $n_i(k+1)$, $\bar{p}_i(k+1)$ for $i \in \mathcal{N}$.
7: **end for**

The performance of Algorithm 1 with varying file sizes (assuming bounded file size) is measured by its *regret* $\mathbb{E}[R(K)] \triangleq \mathbb{E}\sum_{k=1}^{K}(T(\pi^k, F^k) - T(\pi_{tar}(F^k), F^k))$, which is defined as the cumulative difference of expected transfer time between policy π^k at k-th file and the targeted optimal policy $\pi_{tar} \in \{\pi^*, i_{so}, i^*, \pi_{heu}\}$ up to the K-th file. The regret analysis is the nearly same as Theorem 5.4 in [23] and we only present the regret bound of Algorithm 1 due to the page limit. We have,

$$\mathbb{E}[R(K)] \leq \mathcal{O}\left(\frac{NF_{\max}}{\Delta_{\min} p_{\min}^2 r_{\min}} \log(K)\right), \tag{8}$$

where $F_{\max}$ is the largest possible file size, $p_{\min} = \min_{i \in [N]} p_i$, $r_{\min} = \min_{i \in [N]} r_i$ and $\Delta_{\min} = \min_{\{F \in (0, F_{\max}], \pi \in \Pi(F)/\pi_{tar}\}} \mathbb{E}[T(\pi, F)] - \mathbb{E}[T(\pi_{tar}(F), F)]$ is the smallest non-zero difference of expected transfer time between any sub-optimal policy π and the targeted optimal policy π_{tar}. The regret (8) scales linearly with the number of channels N, instead of the number of edges in the online shortest path problem [23], because each edge in our setting (see Figure 3) is chosen from one of N channels while each edge in [23] is treated as a different 'arm'.

VI. Numerical Results

In this section, we present numerical results for file transfer time under four different policies in online settings (unknown p_i's), using three different channel scenarios as in [12], [14], [25]. Through these results, we show the significant time reduction achieved by the dynamic optimal, static optimal and heuristic polices over the max-throughput channel, in line with theoretical analysis.

We consider the experimental setup as an IEEE 802.22 system with 8 different channels. The time duration Δ is set to 100 ms, per IEEE 802.22 standard [3]. We use three different channel scenarios: *gradual, steep* and *lossy* [12], [14], [25]. *Gradual* refers to a case where the available probability of the max-throughput channel is larger than 0.5. *Steep* is characterized by the available probability of each channel being either very high or very low. *Lossy* means that the available probability of the max-throughput channel is smaller than 0.5. The channel parameters in the above three channel scenarios are given in Table I.

TABLE I
Channel parameters in three channel scenarios

channel i	1	2	3	4	5	6	7	8
r_i (Mbps)	1.5	4.5	6	9	12	18	20	23
p_i (*gradual*)	0.95	0.85	0.75	0.65	0.4	0.3	0.2	0.1
p_i (*steep*)	0.9	0.25	0.2	0.18	0.17	0.16	0.15	0.14
p_i (*lossy*)	0.9	0.8	0.7	0.4	0.3	0.25	0.2	0.1

In our setting, the file size need not be fixed. Since larger file sizes naturally take more time to transmit, it makes sense to normalize our performance metric across the range of file sizes. We define our metrics as *average time ratio* and *average throughput*. For an arbitrary sequence of files $\{F^k\}_{k \in \mathbb{Z}_+}$, the average time ratio at the K-th episode is defined as $\frac{1}{K}\sum_{k=1}^{K} T(\pi^k, F^k)/\mathbb{E}[T(i^*, F^k)]$ and the average throughput is represented as $\frac{1}{K}\sum_{k=1}^{K} F^k/T(\pi^k, F^k)$. Here, $T(\pi^k, F^k)$ is the measured transfer time of a file of size F^k applying the policy π^k at the k-th episode. Policy π^k is based on the estimated parameter, which is updated by the SU on the fly, as described in Section V-B.

In our simulation, we generate 7000 files from $(0, 7]$ (Mb) uniformly at random to be used in Algorithm 1. The simulation is repeated 200 times. We observe in Figure 4 that the max-throughput policy achieves the largest average throughput while, counter-intuitively, has the longest transfer time in all channel cases except the lossy case. The reason is that the max-throughput policy computed by the SU is affected by the estimated parameters and can be the inferior policy, resulting in lower average throughput initially. For the lossy case on the right column in Figure 4, even though the red curve (max-throughput policy) is below the blue one for now, which is an effect of imperfect knowledge of channel parameters, we infer that the red curve will eventually exceeds all other curves as the SU will perfectly learn all the channel parameters.

Next we focus on the average time ratio in the left column of Figure 4. We first observe that all curves eventually flatten out, signifying the convergence of Algorithm 1. In the gradual case, the average time ratio is above 95% for all three policies, implying that they don't obtain much reduction in time and the max-throughput channel is good to access when it is available for most of the time. However, as shown in the *steep* and *lossy* cases respectively, the dynamic optimal policy and heuristic policy, as well as the static optimal policy, can save over

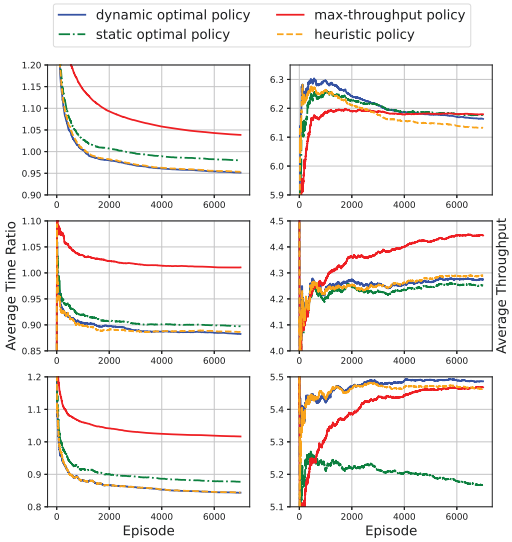


Fig. 4. Average time ratio (left column) and average throughput (right column). Channel scenarios from top to the bottom: Gradual; Steep; Lossy.

10% time on average over the baseline. This observation is in line with Corollary 3.3 and Corollary 4.3 since the available probabilities of the max-throughput channel are very small in *steep* and *lossy* cases. Furthermore, the heuristic policy, in addition to keeping the complexity low, achieves similar transfer time to that of the dynamic optimal policy; at the same time performing better than the static optimal policy, as expected from Section V-A.

VII. Conclusion and Future Work

In this paper, we have developed a theoretical framework for file transfer problem, where channels are modeled as independent Bernoulli process, to provide the accurate file transfer time for both static and dynamic policies. We pointed out that the max-throughput channel does not always minimize the file transfer time and provided static optimal and dynamic optimal policies to reduce the file transfer time. Throughout our analysis, we demonstrated that our approaches can obtain significant reduction in file transfer time over the max-throughput policy for small file sizes or when the max-throughput channel has very high rate but with low available probability, as typically the case in reality. Our future works include the extension to heterogeneous and *Markovian* channels for minimum expected file transfer time, for which our SSP formulation for online learning scenario becomes no longer applicable.

References

[1] Y. Xu, A. Anpalagan, Q. Wu, L. Shen, Z. Gao, and J. Wang, "Decision-theoretic distributed channel selection for opportunistic spectrum access: Strategies, challenges and solutions," *IEEE Commun. Surv. Tutor.*, vol. 15, no. 4, pp. 1689–1713, 2013.

[2] FCC, "Fcc increases unlicensed wireless operations in tv white spaces," Dec 2020. [Online]. Available: https://www.fcc.gov/document/fcc-increases-unlicensed-wireless-operations-tv-white-spaces-0

[3] "Ieee standard - information technology-telecommunications and information exchange between systems-wireless regional area networks-specific requirements-part 22," *IEEE Std 802.22-2019*, pp. 1–1465, 2020.

[4] Q. Zhao, L. Tong, A. Swami, and Y. Chen, "Decentralized cognitive mac for opportunistic spectrum access in ad hoc networks: A pomdp framework," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 3, pp. 589–600, 2007.

[5] Y. Liu and M. Liu, "An online approach to dynamic channel access and transmission scheduling," in *Mobihoc'15*, 2015, pp. 187–196.

[6] Q. Zhao, S. Geirhofer, L. Tong, and B. M. Sadler, "Opportunistic spectrum access via periodic channel sensing," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 785–796, 2008.

[7] K. Liu and Q. Zhao, "Indexability of restless bandit problems and optimality of whittle index for dynamic multichannel access," *IEEE Trans. Inf. Theory*, vol. 56, no. 11, pp. 5547–5567, 2010.

[8] Y. H. Jung and A. Tewari, "Regret bounds for thompson sampling in episodic restless bandit problems," in *NIPS'19*, vol. 32, 2019.

[9] C. Tekin and M. Liu, "Online learning in opportunistic spectrum access: A restless bandit approach," in *INFOCOM'11*, 2011, pp. 2462–2470.

[10] W. Dai, Y. Gai, and B. Krishnamachari, "Efficient online learning for opportunistic spectrum access," in *INFOCOM'12*, 2012, pp. 3086–3090.

[11] L. Lai, H. El Gamal, H. Jiang, and H. V. Poor, "Cognitive medium access: Exploration, exploitation, and competition," *IEEE Trans. Mobile Comput.*, vol. 10, no. 2, pp. 239–253, 2010.

[12] R. Combes, A. Proutiere, D. Yun, J. Ok, and Y. Yi, "Optimal rate sampling in 802.11 systems," in *INFOCOM'14*, 2014, pp. 2760–2767.

[13] H. Gupta, A. Eryilmaz, and R. Srikant, "Low-complexity, low-regret link rate selection in rapidly-varying wireless channels," in *INFOCOM'18*, 2018, pp. 540–548.

[14] ——, "Link rate selection using constrained thompson sampling," in *INFOCOM'19*, 2019, pp. 739–747.

[15] O. Avner and S. Mannor, "Multi-user communication networks: A coordinated multi-armed bandit approach," *IEEE/ACM Trans. Netw.*, vol. 27, no. 6, pp. 2192–2207, 2019.

[16] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 4, pp. 3098–3130, 2018.

[17] H.-P. Shiang and M. Van Der Schaar, "Queuing-based dynamic channel selection for heterogeneous multimedia applications over cognitive radio networks," *IEEE Trans. Multimed.*, vol. 10, no. 5, pp. 896–909, 2008.

[18] Y. Wu, F. Hu, S. Kumar, Y. Zhu, A. Talari, N. Rahnavard, and J. D. Matyjas, "A learning-based qoe-driven spectrum handoff scheme for multimedia transmissions over cognitive radio networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 11, pp. 2134–2148, 2014.

[19] I. Dimitriou and N. Pappas, "Stable throughput and delay analysis of a random access network with queue-aware transmission," *IEEE Trans. Wirel. Commun.*, vol. 17, no. 5, pp. 3170–3184, 2018.

[20] H. Cao, H. Tian, J. Cai, A. S. Alfa, and S. Huang, "Dynamic load-balancing spectrum decision for heterogeneous services provisioning in multi-channel cognitive radio networks," *IEEE Trans. Wirel. Commun.*, vol. 16, no. 9, pp. 5911–5924, 2017.

[21] X.-L. Huang, X.-W. Tang, and F. Hu, "Dynamic spectrum access for multimedia transmission over multi-user, multi-channel cognitive radio networks," *IEEE Trans. Multimed.*, vol. 22, no. 1, pp. 201–214, 2019.

[22] G. V. Rossi and K. K. Leung, "Optimised csma/ca protocol for safety messages in vehicular ad-hoc networks," in *ISCC'17*, 2017, pp. 689–696.

[23] M. S. Talebi, Z. Zou, R. Combes, A. Proutiere, and M. Johansson, "Stochastic online shortest path routing: The value of feedback," *IEEE Trans. Automat. Contr.*, vol. 63, no. 4, pp. 915–930, 2017.

[24] S. Tu and B. Recht, "The gap between model-based and model-free methods on the linear quadratic regulator: An asymptotic viewpoint," in *COLT'19*, 2019, pp. 3036–3083.

[25] J. C. Bicket, "Bit-rate selection in wireless networks," Master's thesis, Massachusetts Institute of Technology, 2005.

[26] D. Bertsekas, *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.

[27] A. Mantuano, "File size basics," 2016. [Online]. Available: https://techdocs.blogs.brynmawr.edu/5523

[28] J. Hu, V. Doshi, and D. Y. Eun, "Opportunistic spectrum access: Does maximum throughput minimize file transfer time?" [Online]. Available: https://people.engr.ncsu.edu/dyeun/pub/osa-delay-techrep.pdf

[29] Y. Gai, B. Krishnamachari, and R. Jain, "Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1466–1478, 2012.

[30] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *ICML'13*, 2013, pp. 151–159.