Exploiting Channel Diversity in White Spaces

Božidar Radunović, Alexandre Proutiere, Dinan Gunawardena, Peter Key

April 2011

1. INTRODUCTION

"White Space" spectrum refers to unused parts of the TV/UHF spectrum (unallocated or not used locally). The FCC 2008 ruling to allow unlicensed devices to use parts of this spectrum has catalyzed research and development in this area. As a part of the 2010 ruling [1], FCC mandates the use of a geo-location database to identify which frequencies are free from primary users. By querying the geo-location database, we are guaranteed to obtain a set of channels free from primary transmitters (e.g. TVs and wireless mics) and we avoid the difficult problem of sensing primary users.

The number of channels available for unlicensed use may be large. In [2] the authors analyze the number of TV channels available across US. The actual number of channels available varies from 5 to 10 on the coasts up to 40 in the Mid-West, with half of the US population having more than 20 TV channels available for white-space communication. Due to variability of the environment, the quality of available channels (in terms of the achievable data rates) can vary significantly and randomly over time and across channels and different links. Hence tracking and transmitting on the *best* channel can greatly improve the system performance. The main challenge that we address in this paper is to dynamically select the best channel and coding rate for transmission.

As previous works demonstrate [3] and our measurements confirm for TV bands, RSSI (Receive Signal Strength Indicator) is a poor predictor of channel quality. In order to accurately estimate a wide-band channel, more sophisticated techniques with specific hardware support are needed, such as those used in [4, 5]. These techniques are not supported in the current commercial white-space radios (e.g. [6, 7]). Instead, we need to infer the quality of each channel at each transmission rate through probing. Single packet probing is not enough, and we need to send several packets on each channel and at each rate to enable us to construct a reliable estimate of the channel quality. In the design of channel and rate selection schemes, we face a classical exploration vs. exploitation trade-off problem. We need to exploit the best (channel, rate) pair, whilst constantly exploring whether this best pair changes over time. Exploration induces a cost since it involves exploring bad (suboptimal) channels and rates. Note that the joint channel and rate selection problem we address is considerably more difficult than merely detecting a channel with no primary user (as in [8]). When detecting primaries, it is sufficient to find any free channel, whereas we search for the best channel.

Our problem is similar to that of rate adaptation in WLANs (c.f. [9–11]), but with key differences. The first difference is that the design space has two dimensions, the channel and the transmission rate. For each packet, we have to select the transmission rate and also the channel on which the transmission should take place. As a consequence, the decision space (channels×rates) is much larger than with WLAN, and the designed channel and rate selection algorithm should learn channel conditions quickly. Note that the number of available channels in white spaces may be much larger than in WiFi (WiFi has 3 orthogonal channels in 2.4GHz band). We have 11 channels and 3 rates in our test-bed and the exploration of all (channel, rate) pairs a substantial challenge. Fortunately, the goodputs obtained by transmitting on the same channel but at different rates are inherently correlated, which can be used to speed up the exploration phase.

The second key difference is that in WLANs, the rate adaptation algorithms [9–11] operate on a single channel. They probe the channel in each transmission, and hence are able to continuously monitor the channel quality. When multiple channels are available, one can only acquire information about the quality of the currently used channel, and only get sporadic information about the other channels when they are probed. An important component of our problem is to decide when and how often to explore the other channels.

An additional difficulty stems from the unpredictable evolution of channel conditions. Our measurements reveal *non-stationary* channel conditions, i.e., the success probability of a transmissions using a given (channel, rate) pair varies over time in an unpredictable manner, whereas most of the related work in this area [12–18] assumes that the stochastic processes capturing the evolutions of channel conditions are *stationary* (the most common assumption is that the channel can be modeled as a Markov chain).

A naive way of designing channel and rate selection schemes is to use the (channel, rate) pair that is believed to yield the greatest goodput, and periodically probe all other pairs to detect significant channel condition changes. However, as our measurements show, channels change continuously and often independently from each other. Deciding when and how often to stop exploiting the best (channel, rate) pair to probe other alternatives is one of the main difficulties of the problem.

In this paper, we consider a down-link scenario, where an access point sends data to clients. We design and implement a novel channel and rate selection and scheduling algorithm for this scenario. The access point uses information on the outcome of previous transmissions to continuously learn channel qualities, and proactively adapts channel and rate. The proposed algorithm is based on a Multi-Armed Bandit (MAB) learning approach and builds on the existing MAB algorithms [19, 20]. We show that it performs well in the realistic, non-stationary channel conditions, and that it provides significant gain over channel-agnostic approaches.

The main contributions of the paper are as follows:
• A measurement study in an office White-Space testbed operating in the 500MHz to 600MHz band. From these measurements we quantify the possible goodput gains achieved by always tracking and using the best (channel, rate) pair, compared to simple fixed channel strategies. We observe substantial possible gains (up to 100%).
• A novel, joint scheduling and learning algorithm to dynamically select users, channels and rates for transmission,

called FSS-UCB[1] (Section 5). Our algorithm extends the classical UCB algorithm [19, 20]: (i) it is adapted to non-stationary environments; (ii) it uses the notion of *Soft Sampling* introduced to leverage correlations between goodputs obtained on the same channel at different rates, and to alleviate the problem of having many (channel, rate) pairs to explore; (iii) it incorporates opportunistic sampling to further speed up the learning across the links and (iv) it provides fair sharing among the users.

• A demonstration of performance gains. We implement our algorithms in the test-bed, and explain how to solve synchronization problems when switching channels (Section 7). We quantify the performance of the algorithms in both single link and Access-Point scenarios (Section 8). They achieve a goodput improvement ranging from 40% to 90% over fixed channel schemes and also improve fairness in the AP scenario.

2. RELATED WORK

There is a lot of recent research on the design of opportunistic channel selection algorithms for cognitive radio systems, exploiting, for example, Whites Space spectrum, see e.g. [14–16, 21, 22]. Similarly, rate adaptation has triggered considerable research effort, see e.g. [5,9,10]. To our knowledge, the present paper is the first attempt to design and implement dynamic joint channel and rate selection algorithms.

Detecting primary users: Most of studies on channel selection in cognitive radio networks deal with learning algorithms able to find and exploit channels free from primary users, as in [8, 12–15, 23, 24]. This means that each channel is assumed to be either in a "bad" or "good" state. This model does not apply to our scenario because it suffices to find any "good" channel, whereas in our case we need to find the best channel and rate pair among all pairs.

We do not consider the problem of detecting primary users. Recent FCC proposals [1] mandates the use of geo-location databases to help cognitive users avoid TV signals. Some countries use database approach to protect wireless mics (e.g. [25] in UK).

Opportunistic scheduling in cellular networks: Opportunistic scheduling has been initially proposed for cellular network [26], and has been implemented in recent cellular standards (e.g. HDR and HSDPA). Before transmitting a base station probes the channels to all registered users and selects the one based on the channel state (for efficiency) and previously offered service (for fairness). However, existing opportunistic schedulers in cellular do not allow for learning about multiple channels.

Probing before transmission: Several papers propose extension of opportunistic scheduling to multi-channel systems [21, 27, 28]. The common ground for all these works is that channels are probed with small packets before the transmission. The issue is how to find the right trade-off between the overhead of probing and finding the channel with the highest quality. Probing is not applicable in our design. Channel switching time is too long and the overhead of probing channels (11 in our case) will be an order of magnitude larger than the packet transmission time.

Learning stationary channels: Work such as [12–18] considers the channel learning problem using classical stochastic control techniques, such as Markov Decision Processes (MDP). They assume that the underlying stochastic processes capturing the evolutions of channel conditions are *stationary* with either unknown distributions, or known distributions but unknown realizations. In other words, they assume that the packet transmission success probabilities do not vary over time, and that they can be even known in advance. In practice this assumption does not hold, as we verify in Section 3. Due to mobility in the environments, wireless transmitters face non-stationary, arbitrarily changing, and unknown channel conditions.

Learning non-stationary channels: Learning in a non-stationary environment has been formulated in the learning theory community as a non-stationary multi-armed bandit problem (MAB) [19, 20, 29]. We are not aware of any work applying non-stationary channel models to our context of channel and rate learning. Our work is most similar to [20], with three main distinctions. We introduce soft-sampling as a mean to speed up the learning process, which otherwise would not be applicable to our problem. We also propose a joint utility scheduling and learning algorithm that exploits broadcast nature of wireless to speed up the learning, and we offer fairness.

Rate adaptation: Several authors (e.g. [9–11]) consider rate adaptation as a function of previously observed transmission successes and losses. These approaches do not apply to our problem for two reasons. Firstly, in [9–11] if one rate is repeatedly successful, we can move to the next higher rate. This does not hold for different channels, since they are uncorrelated (as we demonstrate in Section 3).

Recently, several papers have proposed the use of advance PHY support for channel estimation and rate adaptation [4, 5]. We are interested in channel and rate selection algorithm that will work with future white space hardware that will possibly use off-the-shelf WiFi hardware and frequency transposer (such as [6, 7]). We thus assume we dispose of a standard OFDM PHY, and the techniques from [4, 5] do not apply.

Channel assigment in WiFi: The channel assignment problem in WiFi networks has been well studied (see e.g. [30] and the references therein). Most of these focus on power control and/or orthogonal channel assignment between interfering clients to minimize channel contention. Due to their underlying complex optimization component, these algorithms take long time to converge. For example, utility assessment intervals in [30] last about 300s. In contrast, our algorithm can estimate a single channel in less than 10s (see the discussion in Section 7.1), and the mean time of channel

---

[1]FSS stands for Fair, Soft-Sampling, and UCB for Upper Confidence Bound
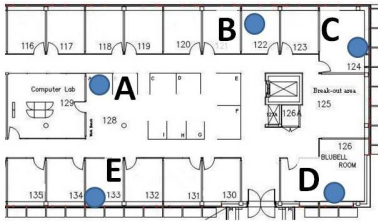
Figure 1: Our testbed. Node A is an access point and nodes B, C, D and E are clients.

change observed in our measurements is about 95s (Observation 6),

Decentralized opportunistic channel selection: A more general problem that can be posed is how to select the best channel and rate pair while taking into account the number of users also using that channel. Intuitively, one may prefer a lightly loaded channel over a highly loaded channel even if the quality of the highly loaded channel is better. This is a difficult and open problem. It has been recently studied in [31–33], in a very theoretical context.

3. TESTBED AND MEASUREMENTS

We used an SDR platform in an indoor office testbed to take initial measurements to guide the design of our channel and rate selection algorithms. The testbed is also used to implement and test our algorithms. In this section we describe the platform, the measurements and the potential gains that can be achieved in practice by tracking the best (channel, rate) pair for transmission.

3.1 Testbed and Measurements Setup

Our indoor testbed uses nodes located on the same floor of an office building, shown in Figure 1, comprising a mix of open-plan space and offices. Most of the fixed barriers are glass, wood, or concrete, which reflect and attenuate radio signals, while human mobility patterns added significant temporal variability. The nodes are based on the Lyrtech SFF-SDR platform that has a single radio with separate transmit and receive circuits, each with its own antenna.

We operate the board in the UHF band, between 500MHz and 610Mhz (for which we received an experimental indoor license). Throughout our experiments we have not observed any noticeable interference from other transmissions (such as TVs).

We use OFDM as the physical (PHY) layer. The PHY is very similar to 802.11a/g, except that it has 10 MHz bandwidth instead of 20MHz. The 10MHz bandwidth approximately corresponds to the bandwidth of TV channels, which in most countries is 6-8 MHz. We use QPSK modulation, and codes of respective rates 1/2, 2/3 and 3/4. This allows us to transmit packets at 3 different rates: 4.5Mbps, 6Mbps and 6.75Mbps[2]. Channel switching time takes the equivalent of

---

[2]We have also experimented BPSK modulation but in all cases it proved to yield lower goodput, so we do not report these results

1 or 2 packet transmission times.

In our experiments, one board is the access point (AP) used to broadcast packets. It is placed in the open plan area (Figure 1 label A). Four other boards, located in four different offices (B,C, D and E), act as clients which receive and log all packets received from the AP. The measurements are taken during the daytime with plenty of people in the open plan area in front of transmitter A.

In each experiment, unless otherwise specified, the AP transmits 10 packets at each rate on a given channel before moving to the next channel. 11 channels of width 10Mhz (between 500-610MHz) are probed, and hence each measurement round consists of $10 \times 11 \times 3 = 330$ packets. Packet transmissions are made periodically every 5ms so that round duration is less than 2s. This means we sample a given channel at a given rate once every 2s. In each round and for each channel, we calculate, by averaging over 10 packets, the RSSI, the success packet transmission probabilities and the goodputs[3] at the 3 different rates.

The AP dictates the channel the other nodes need to listen to. To maintain synchronization, we encode in each transmitted packet header the channel the AP will use next. With high probability all four nodes receive at least one of the 30 transmitted packets on each channel and stay in sync. When a node fails to receive all such packets, it simply waits until the AP loops back to the channel it was listening to.

3.2 Channel Measurements

We now report on our measurements. It turns out that the average channel conditions vary slowly over time, and to be hardly correlated across channels. We also observe that the success transmission probabilities at different rates cannot be well predicted by just measuring the RSSI. Finally, the measurements suggest that tracking the best (channel, rate) pair for transmission may yield significant performance improvements. We now describe in detail these observations.

Fast and slow fading

We first look at the effect of the fast and slow fading. In Figure 2 (a), we present a snippet of the evolution of the goodputs over time obtained on two channels when transmitting at different rates. We observe some variability in the goodput from one batch to the other. This is the result of *fast fading* whose time-scale (coherence time) is smaller than a few seconds, which is our measurement interval. However, by averaging over a few consecutive batches, the goodput obtained with a given (channel, rate) pair is stable.

Observation 1: Our hardware is too slow to learn and adapt to fast fading. Instead, we should treat fast fading as a random process and try to learn and track its average.

We also observe slow channel variations. Such variations are referred to as *slow fading*, and typically caused by hu-

---

here.

[3]The goodput is defined as the product of the success transmission probability and of the transmission rate.
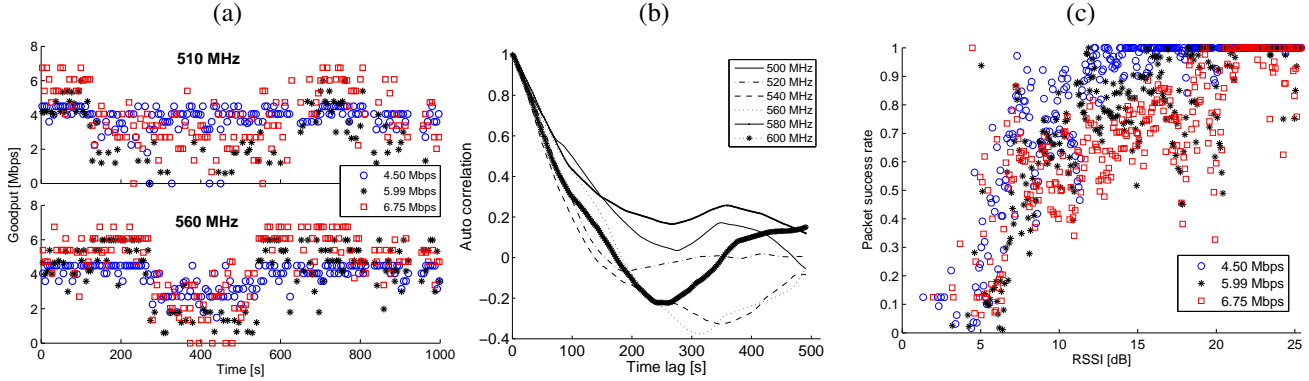
Figure 2: (a) The goodput (transmission rate $\times$ average success rate) of each batch of 10 consecutive packets for two selected channels and transmission rates for link A-C; (b) The auto-correlation of the average rate as a function of a time lag for different rates and channels for link A-C; (c) Packet success rate at different transmission rates as a function of RSSI (normalized over the initially observed noise).

man mobility and related changes in the environment. Visually, from Figure 2 (a), it seems that significant changes can be observed at a time-scale of the order of several tens of seconds. We quantified this claim by evaluating the auto-correlation[4] function of the goodputs of each (channel, rate) pair. We plot the auto-correlation in Figure 2 (b). We observe that the auto-correlation at lags smaller than 30s was always greater than 80%, hence a typical duration of a slow fade is of order of tens of seconds or more. In our experiments, it seemed that slow fading exhibits much larger variations than fast fading (see e.g. Figure 2 (a) – channel at 560MHz).

Observation 2: The slow fading time-scale is of the order of tens of seconds or more. During slow fades, channel changes are larger than during fast fades, thus it is more important to track slow fading than fast fading.

Correlations across channels

Our measurements showed a low correlation between qualities of various channels for a given link at a given rate. We found that cross-correlation[5] of goodputs on any two channels at a given rate was always smaller than 30%. This observation is in accordance with theory [26, eq.(2.47)]: Given the size of our test-bed, the expected delay spread is of order of tens of nano-seconds, and the coherence frequency is of order of a few MHz. Hence we expect that each 10MHz-wide channel will not be correlated with the adjacent ones.

Observation 3: Slow fades do not occur at the same time at all channels. This lack of correlation highlights the importance of tracking the quality of *all* channels so as to enable us to always select the best for transmission.

---

[4]The auto-correlation at lag τ of a sample time series $x_1, \ldots, x_N$ is defined by : $\sum_{t=1}^{T}(x_t - \bar{x})(x_{t+\tau} - \bar{x}) / \sum_{t=1}^{T}(x_t - \bar{x})^2$ where $\bar{x} = \sum_{t=1}^{N} x_t / N$ and $T < N$ is a large number.

[5]The cross-correlation of two time series $x_1, \ldots, x_N$ and $y_1, \ldots, y_n$ is defined by : $\sum_{t=1}^{N-\tau}(x_t - \bar{x})(y_t - \bar{y}) / (\sum_{t=1}^{N}(x_t - \bar{x}) \sum_{t=1}^{N}(y_t - \bar{y})$ where $\bar{x} = \sum_{t=1}^{N} x_t / N$ and $\bar{y} = \sum_{t=1}^{N} y_t / N$.

Rate as a function of RSSI

We next evaluate whether RSSI can be use as an indication of channel quality. We divide the RSSI scale into 500 equally sized bins and classify all packets recorded during our measurement according to the observed RSSI into one of the bins. We calculate the aggregate packet transmission success rate for each of the bins and report the results in Figure 2 (c). We have the following observation.

Observation 4: Although there is a general correlation between the RSSI and packet transmission success rate, RSSI does not give us accurate information on the channel quality.

We note that this observation is consistent with previously reported observations in 2.5 GHz band [3, 4].

Success transmission probabilities at different rates

In Figure 3, we graph the conditional success transmission probabilities at two different rates given the success probability of transmissions at the third rate. To obtain these results, the AP sent 3 consecutive packets using the 3 different rates and did this 1000 times which defines a round. Each point in Figure 3 corresponds to the success transmission probabilities averaged over a round. We plot points for all 11 channels and all 4 destinations.

From Figure 3 we can conclude that there is indeed a clear ordering between rates, as we would expect, in that lower rates have higher success probability. However, it also shows that it is difficult to infer the packet success rate at one rate when probing on a different rate.

From the above two remarks, we deduce that to track the best (channel, rate) pair we actually need to explore all the possible rates. Nevertheless, to speed up the exploration, we may leverage the fact that higher rates exhibit higher packet loss probabilities. In the design of channel and rate selection algorithms, we actually exploit the following observation.

Observation 5: It is difficult to infer the packet transmission success rate at one rate when probing at a different rate. However, (a) if a packet transmission is successful at a given rate on a given channel, then it would have been successful
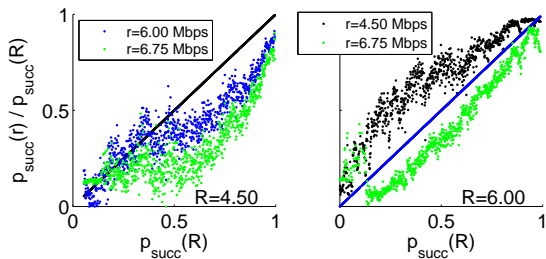
Figure 3: Conditional probability of successful transmission at different rates, given the success probability at rate R, averaged over 1000 consecutive packets.
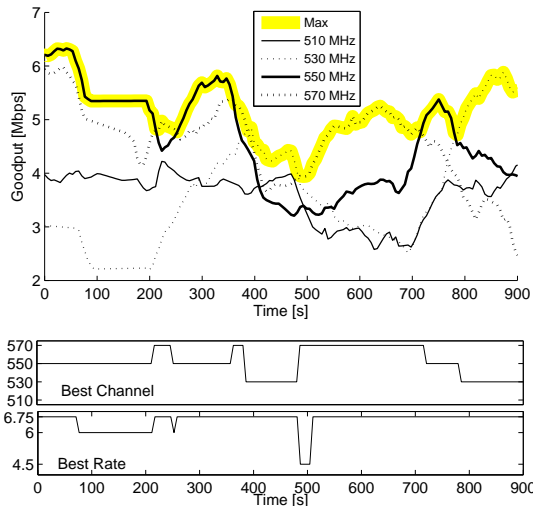

Figure 4: (Top) Goodput achieved by selecting the best rate on various channel for link A-C. (Bottom) Evolution in time of the best (channel, rate) pair.

at a lower rate on the same channel. (b) Similarly, if a packet transmission fails at a given rate, then it would have failed at a higher rate.

Gains from selecting the best (channel, rate) pair

We conclude this section by quantifying the gain in terms of goodputs that can be achieved by tracking the best (channel, rate) pair. Figure 4 illustrates the performance of the best channel and rate, where only 5 channels are shown for clarity. Observe that the goodput differs widely from one channel to another. In Figure 4 (top), we compared the achieved by always using the same channel at the best rate with that of an Oracle algorithm dynamically selecting the best (channel, rate) pair. The average gains over the algorithm operating on the worst channel is about 80-90% and the gains over an algorithm operating on a randomly selected channel is about 40-50%. This suggests that tracking the best channel and rate can bring substantial performance benefits.

Figure 4 (bottom) illustrates how the best (channel, rate) pair varies over time. It further shows that when transmitting on a given channel, tracking the best rate is important (this is also confirmed in Figure 2 (a)). Finally, we found that the time interval during which a channel remains the best,

over the entire measurement campaign, has mean duration 94.6s and standard deviation 104.3s. As we shall demonstrate, these relatively long periods during which the best channel remains the same facilitates the design of learning algorithms able to really track the best channel, even when a large number of channels have to be explored (11 in our case).

Observation 6: Significant performance gains may be achieved by selecting the best channel and rate for transmission. Typically the periods of time where a given channel is optimal have quite large durations (tens of seconds), which simplifies the design of learning algorithms tracking the best channel for transmission.

4. PROBLEM DEFINITION

We want to design dynamic channel and rate selection schemes that efficiently track the best (channel, rate) pair for transmission. We consider two network scenarios: (i) A single link. For this scenario, the objective is to maximize the long-term goodput of the link. (ii) Single link scenario is a special case of a downlink scenario where an AP serves multiple fully backlogged clients. Here, fairness among the various clients has to be considered. With this aim, we use the classical notion of *utility*. More precisely, let U denote an increasing and concave utility function. The objective is now to maximize the social welfare defined by $\sum_u U(\phi_u)$ where ϕ_u represents the long-term goodput achieved by client u. If $U = \log$, we target a Proportional Fairness (as in Qualcomm's HDR system). With multiple clients, our algorithms have to dynamically select not only a (channel, rate) pair but also a client for transmission.

We assume that the set of available channels (i.e., free from primaries) is known and does not change over time. As mentioned earlier, RSSI is not a sufficiently accurate predictor of the channel quality (Observation 4). We also assume that our radio devices do not support advanced PHY techniques for channel estimation, such as those described in [4, 5]. Furthermore, probing before each transmission is too costly. Instead, we focus on algorithms that learn about channel quality from past observations of transmission successes.

Given the number of channels and rate, our system cannot track fast fading (Observation 1). However, we can track slow fading (Observation 2). Note that the performance benefits from opportunistically exploiting slow fading are substantial (Observation 6), and higher than those we would obtain by exploiting fast fading (Observation 2).

As explained earlier, the main challenge in the design of channel and rate selection schemes stems form the fact that we cannot continuously monitor the evolution of the quality of *all* channels, since the AP uses a single channel for transmission at a given time. Channel conditions change in an uncorrelated manner (Observation 3), and hence we need to explore other (channel, rate) pairs often to detect opportunities to improve the system performance. The exploration is

complicated by the large number of possible (channel, rate) pairs. In our case we have 11 channels × 3 rates = 33 pairs, and exploring all possible (channel, rate) pair could well be too costly. Fortunately, we may exploit the inherent correlations between the transmission success probabilities when using different rates but the same channel.

5. JOINT CHANNEL, RATE SELECTION AND SCHEDULING ALGORITHM

In this section, we propose FSS-UCB, a learning algorithm to jointly and dynamically select channels, rates and clients for transmission. We first describe the different components of the algorithms, and then provide their pseudo-codes, given in Table 1.

5.1 Notation

Consider an access point (AP) serving N fully backlogged clients indexed by $u = 1, \ldots, N$. Transmissions may be carried on C different channels at L different rates $r_1 < \ldots < r_L$ depending on the chosen modulation and coding scheme. In our test-bed $C = 11$ and $L = 3$.

Time is divided into short *frames* of constant duration during which the AP transmits a few packets to the same client using the same (channel, rate) pair. Let m_l denote the number of packets that can be sent at rate r_l during a frame. It represents the rate normalized by the frame duration (note that $\frac{m_l}{m_k} = \frac{r_l}{r_k}$ for all k, l). Without loss of generality, we assume that frames have unit duration. At the beginning of each frame, the AP selects a (channel, rate) pair and a client for all the transmissions in this frame. This decision is based on the past observations only. The use of fixed duration frames allows periodic updating which keeps the algorithm simple (decision updates taken after every packet transmission would require to take into account packet transmission durations and would make the algorithms more complex). We index frames with index t, and packets within each frame with index $p = 1, 2, \cdots$.

During frame t, if user u is served on channel i and rate r_l, the packet transmissions are successful with probability $\mu_{u,i,l}(t)$. This probability is unknown and has to be learned. It is non-stationary, and changes over time.

Our algorithms keep track of two variables that store information about past transmissions. The first one is the the empirical discounted *number of frames* $\hat{n}_{u,i,l}(t)$ we have used to transmit to user u on channel i and at rate r_l up to frame t. The second one is the empirical discounted *number of successfully transmitted packets* $\hat{x}_{u,i,l}(t)$ for each tuple (u, i, l) up to frame t. The variables are updated at the end of each frame. From the two set of variables, we deduce the estimated average success probability $\hat{\mu}_{u,i,l}(t)$ up to frame t of transmissions to client u on channel i and at rate l as

$$\hat{\mu}_{u,i,l}(t) = \frac{\hat{x}_{u,i,l}(t)}{m_l \hat{n}_{u,i,l}(t)}.$$

5.2 Exploration vs. Exploitation

Most of the time we want to use the best channel and rate for transmission. However, we also need to occasionally explore the other channels and rates, to detect better transmission opportunities.

We build on the approach from [19, 20] in order to achieve a provably optimal balance between exploitation and exploration (for detailed analysis, see Section 6). For each frame t, each link u and each (channel, rate) pair (j, k) are assigned a weight $w_{u,j,k}(t)$. In case link u is selected for transmission at time t, we then select the pair (j, k) with the highest weight. The weight

$$w_{u,j,k}(t) = \max \left(m_k, \left[\frac{\hat{x}_{u,j,k}(t)}{\hat{n}_{u,j,k}(t)} \right. \right.$$
$$\left. \left. + \xi \, m_L \sqrt{\frac{\log(\sum_{i,l} m_l \hat{n}_{u,i,l}(t))}{m_k \hat{n}_{u,j,k}(t)}} \right] \right) \quad (1)$$

is composed of two terms defining the exploration vs. exploitation trade-off. The first term is the observed average goodput $\frac{\hat{x}_{u,j,k}(t)}{\hat{n}_{u,j,k}(t)}$ up to frame t. The second term is the exploration factor. As the different (channel, rate) pairs are explored, the enumerator $\log(\sum_{i,l} m_l \hat{n}_{u,i,l}(t+1))$ increases. However, if we do not explore a particular (channel, rate) pair (j, k), the corresponding denominator will remain small. Thus the exploration factors will gradually increase for all (channel, rate) pairs we do not explore and will decrease for those that we explore.

The specific definition of the weights is taken from [19, 20] and it is very important for the performance guarantees of the algorithm. We discuss it in detail in Section 6. However, we introduce here an important modification to the weight calculation as compared to [19, 20]. We initialize the weight for (j, k) at a value m_k, which represents the maximum number of successful transmissions per frame at rate r_k. Furthermore the weight is bounded by m_k. We do this to exploit the fact that if selecting channel i and rate r_l has proven to lead to an average goodput greater than m_k packets per frame for $k < l$, then it is useless to explore rate r_k even if the latter has not been selected at all. So with this choice, we explore first higher rates, and do not explore smaller rates unless this becomes really necessary.

Parameter ξ tunes the balance between the exploitation and the exploration. The higher ξ is, the more aggressively we learn other channels and rates, but the higher the overhead we incur as we use the best currently known channel less often. We discuss the impact of ξ on the performance in Section 7.1.

5.3 Soft-Sampling to exploit side information

The main problem with the exploration from (1) is the large number of (channel, rate) pairs (j, k) we need to explore. To speed up the exploration, we can exploit useful side information. We know from Observation 5 that if a transmission at rate $r_l > r_k$ is successful, transmitting at

rate r_k would also be successful. Similarly, if a transmission at rate $r_l < r_k$ fails, then it would also fail at rate r_k. In particular when $r_l < r_k$, $\mu_{i,l}(t) > \mu_{i,k}(t)$.

Assume that during frame t, a packet p is transmitted to user u on channel i at rate r_l. Let $y_{u,i,l}(t,p)$ be a binary variable indicating whether the transmission is successful ($y_{u,i,l}(t,p) = 1$ in case of success). When sampling (or selecting) channel i and rate l for user u, and observing the outcome $y_{u,i,l}(t,p)$, we will randomly generate corresponding *soft-samples*[6] $y'_{u,i,k}(t,p)$, $k \neq l$, of fictitious transmissions at all other rates, for the same channel i, with the following probability distributions:

Successful transmission: If $y_{u,i,l}(t,p) = 1$, then we generate a soft-sample outcome $y'_{u,i,k}(t,p)$ for all $k \neq l$ as:

$$y'_{u,i,k}(t,p) = \begin{cases} 1, & \text{with prob. } \min\left(1, \frac{\hat{\mu}_{u,i,k}(t)}{\hat{\mu}_{u,i,l}(t)}\right), \\ 0, & \text{with prob. } 1 - \min\left(1, \frac{\hat{\mu}_{u,i,k}(t)}{\hat{\mu}_{u,i,l}(t)}\right), \end{cases} \quad (2)$$

Failure: If $y_{u,i,l}(t,p) = 0$, then we generate a soft-sample outcome $y'_{u,i,k}(t,p)$ for all $k \neq l$ as:

$$y'_{u,i,k}(t,p) = \begin{cases} 0, & \text{with prob. } \min\left(1, \frac{(1-\hat{\mu}_{u,i,k}(t))}{(1-\hat{\mu}_{u,i,l}(t))}\right), \\ 1, & \text{with prob. } 1 - \min\left(1, \frac{(1-\hat{\mu}_{u,i,k}(t))}{(1-\hat{\mu}_{u,i,l}(t))}\right). \end{cases} \quad (3)$$

The intuition behind the above updates can be explained using a very simple model where a hidden random variable $U_{u,i}(t)$, uniformly distributed on [0,1], represents an instantaneous quality of channel i to user u at time t. If $U_{u,i}(t) \leq \mu_{u,i,l}(t)$, then the transmission on channel i to user u at rate l is successful. Suppose we observe a successful transmission at rate l and we want to generate a soft-sample for rate $k > l$. We then know that $U_{u,i}(t) \leq \mu_{u,i,l}(t)$ but we don't know whether $U_{u,i}(t) \leq \mu_{u,i,k}(t)$. The probability that $U_{u,i}(t) \leq \mu_{u,i,k}(t)$ under condition that $U_{u,i}(t) \leq \mu_{u,i,l}(t)$ is indeed $\frac{\mu_{u,i,k}(t)}{\mu_{u,i,l}(t)}$. Similarly, we derive the probability for the other soft-sample updates. In addition, in the actual updates we replace the (unknown) success rates $\mu_{u,i,k}(t)$ with their estimates $\hat{\mu}_{u,i,k}(t)$.

We next provide the way the variables stored by the algorithms are updated. To cope with non-stationary channel conditions, we use a discount factor $\gamma < 1$ in each round to progressively decrease the impact of previous estimates (as done in D-UCB algorithm [20]). Suppose that tuple (u^*, i^*, l^*) was selected by the algorithm for frame t. We then have

$$\hat{n}_{u,i,l}(t+1) = \begin{cases} \gamma \hat{n}_{u,i,l}(t) + 1, & \text{if } (u,i,l) = (u^*, i^*, l^*), \\ \gamma \hat{n}_{u,i,l}(t) + q, & \text{if } (u,i) = (u^*, i^*), l \neq l^* \\ \gamma \hat{n}_{u,i,l}(t), & \text{otherwise.} \end{cases} \quad (4)$$

[6]For the probed rate, the soft sample is by definition equal to the observed sample $y'_{u,i,l}(t,p) = y_{u,i,l}(t,p)$

$$\hat{x}_{u,i,l}(t+1) = \begin{cases} \gamma \hat{x}_{u,i,l}(t) + \sum_p y'_{u,i,l}(t,p), \\ \qquad \text{if } (u,i,l) = (u^*, i^*, l^*), \\ \gamma \hat{x}_{u,i,l}(t) + q \sum_p y'_{u,i,l}(t,p), \\ \qquad \text{if } (u,i) = (u^*, i^*), l \neq l^* \\ \gamma \hat{x}_{u,i,l}(t), \quad \text{otherwise.} \end{cases} \quad (5)$$

We take by definition $\hat{n}_{u,i,l}(0) = 0$ and $\hat{x}_{u,i,l}(0) = 0$. Also notice that soft-samples cannot be treated as real samples, as we do not actually observe the corresponding goodputs. Soft-samples are of lower quality than real samples. To account for this difference, we attach to soft-samples a quality parameter $q < 1$.

The main benefit of soft-sampling is that it allows us to design a learning algorithm as if *all* rates for one channel were sampled at each frame. Thus with soft-sampling, we artificially reduce the number of possible decisions at each frame from $U \times C \times L$ to $U \times C$, which greatly improves the performance.

5.4 Opportunistic Channel Sampling

In the AP scenario, the AP has to serve a (potentially) large number of users. The channel towards each user is different and has to be explored separately, which increases the overhead and may cause the learned channel information to be outdated by the time they get to be exploited.

To circumvent this difficulty, we exploit the broadcast nature of the wireless medium: each transmitted packet can be heard by all users, so a packet sent during frame t to user u on channel i at rate r_l can be used to learn the success rates $\hat{\mu}_{u',i,l}(t)$ for all users u'. More precisely, when transmitting to user u at channel i and rate l during frame t, we record soft samples $y'_{u',i,l}(t,p)$ for each packet p and each user u', and we use all the obtain valued to execute the updates (4) and (5).

We note that the observations $y'_{u',i,l}(t,p)$ are recorded at clients, which in turn need to feed that information back to the AP to update the learning algorithm. We discuss the implementation of the feedback procedure in Section 7.

5.5 Efficiency vs. Fairness

If success rates $\mu_{u,i,l}(t)$ were known at the AP, one way of maximizing social welfare in the long run would be to use a simple gradient algorithm (as in Qualcomm PF scheduler). We define *the discounted throughput* for user u at time t as:

$$\phi_u(t) = \frac{1 - \gamma'}{1 - \gamma'^{t+1}} \sum_{s=0}^{t} \gamma'^{t-s} \sum_{i,l,p} y_{u,i,l}(t,p), \quad (6)$$

for some discount factor $\gamma' \in (0, 1)$.

We then propose the following scheduling algorithm, which consists in selecting for frame t, user u, channel i and rate r_l such that the product of the achieved instantaneous rate and of $U'(\phi_u(t))$ is maximized, i.e.,

$$(u, i, l) = \arg\max_{(u,i,l)} w_{u,i,l}(t) \times U'(\phi_u(t)). \quad (7)$$

where $w_{u,i,l}(t)$ is the estimated rate of the link to user u (augmented with the exploration factor). We discuss the properties of this scheduling algorithm (7) in Section 6.

5.6 Pseudo-code

Finally, we give the pseudo-code of our algorithms in Table 1.

Algorithm FSS-UCB

Initialization: $w_{u,i,l}(0) = m_l$, $\phi_u(0) = 0$, for all i, l, u.
At each time frame $t = 0, 1, 2, \ldots$

1. Select user u_t, channel i_t, and rate r_{l_t} (breaking ties arbitrarily) such that:

$$(u_t, i_t, l_t) = \arg \max_{(u,j,k)} w_{u,j,k}(t)\, U'(\phi_u(t)).$$

   For the entire frame, send packets to user u_t on channel i_t and at rate r_{l_t}.

2. For packets sent during the frame, observe the outcomes $y_{u,i_t,l_t}(t,1), \ldots, y_{u,i_t,l_t}(t,p)$, for all u.

3. For each $k \neq l_t$ and u, generate m_k soft-samples $y'_{u,i_t,k}(t,s)$, $s = 1, \ldots, p$ according to the distributions defined in (2)-(3).

4. Update $\hat{n}_{u,i_t,k}(t+1)$ and $\hat{x}_{u,i_t,k}(t+1)$ for all k, u, using (4) and (5), respectively.

5. Update the discounted throughputs $\phi_u(t+1)$ using (6) for all users u.

6. Update the weights using (1)

Table 1: Pseudo-code of the FSS (Fair, Soft-Sample) UCB algorithm

6. ANALYSIS

In this section we briefly discuss the performance guarantees achieved by our algorithm, FSS-UCB. We first analyze the performance of the learning components of the algorithm, i.e., we consider the single link scenario. Let u be this link. To assess the performance of FSS-UCB, we use the notion of *regret*. The latter is the difference between the goodput achieved by an ideal algorithm always tracking the best (channel, rate) pair with that obtained by FSS-UCB. More precisely, an ideal algorithm would select in frame t the (channel, rate) pair $(i_t^\star, l_t^\star)$ maximizing the expected number of packets successfully sent during this frame:

$$(i_t^\star, l_t^\star) = \arg \max_{(j,k)} m_k \mu_{u,j,k}(t).$$

Now denote by (i_t, l_t) the (channel, rate) pair selected by FSS-UCB. The regret of FSS-UCB up to time T is then:

$$\text{regret}_u^\pi(T) = \frac{1}{T} \sum_{t=0}^{T-1} (m_{l_t^\star} \mu_{u,i_t^\star,l_t^\star}(t) - m_{l_t} \mu_{u,i_t,l_t}(t)). \quad (8)$$

As shown in the following result, the regret of FSS-UCB is controlled by the rate α at which the best (channel, rate) pair $(i_t^\star, l_t^\star)$ changes over time.

PROPOSITION 1. *Assume that the rate at which the best (channel, rate) pair $(i_t^\star, l_t^\star)$ changes over time is bounded by $\alpha > 0$, i.e.,*

$$\limsup_T \frac{1}{T} \sum_{t=1}^{T} 1_{(i_t^\star, l_t^\star) \neq (i_{t-1}^\star, l_{t-1}^\star)} \leq \alpha.$$

Then we have:

$$\limsup_{T \to \infty} \text{regret}^{FSS\text{-}UCB}(T) \leq O(K\sqrt{\alpha}\log(1/\alpha)). \quad (9)$$

PROOF. It is easy to verify that soft-sampling and opportunistic sampling introduced in FSS-UCB still maintain the asymptotic performance properties of the D-UCB algorithm [20] (without soft-sampling and opportunistic sampling). Hence, the proof follows directly from [20, Remark 4]. □

A direct consequence of the above proposition is that when the rate α at which the best (channel, rate) pair changes tends to zero, then the regret of FSS-UCB also converges to zero. In particular, when the best pair does not change, FSS-UCB is a zero regret algorithm as stated in the following corollary.

COROLLARY 1. *If the channel conditions are i.i.d. over time (i.e., the average success rates $\mu_{u,i,l}(t)$ are constant), then:*

$$\limsup_{T \to \infty} \text{regret}^{FSS\text{-}UCB}(T) = 0. \quad (10)$$

This contrasts with a naive strategy that would periodically explore channels and rates, which would incur a constant strictly positive regret even when the channels are stationary. Note also that the performance bound derived in Proposition 1 basically states that FSS-UCB is well suited for tracking the best channel and rate when they change infrequently, as observed in Section 3. Finally observe that, although FSS-UCB has the same *asymptotic* performance guarantees as those of D-UCB from [20], it generally performs much better thanks to the introduction of soft-sampling and opportunistic sampling. Indeed the performance of D-UCB would be quite low in our setting where the number of (channel, rate) pairs is large.

We conclude this section by arguing that for an AP scenario, FSS-UCB will maximize the long-term social welfare of the system. Specifically, when γ' becomes close to 1, and when the frequency of channel changes α becomes close to zero, the gradient algorithm (6)-(7) tends to maximize utility. This statement can be proved using classical stochastic approximation techniques, see e.g. [34]. We omit the proof due to lack of space.

7. IMPLEMENTATION

We implemented our algorithm in the MAC layer of the Lyrtech SFF SDR boards. We use a standard, unmodified OFDM PHY. The algorithm is prototyped on a TI SDP DM6446 DMP in C. Our current implementation, with no optimization, executes a scheduling procedure in less than 100 μs. The total MAC header overhead is 6 bytes.

In what follows we discuss some of the important practical aspects of the channel and rate selection problem that we have encountered during the implementation. We first discuss the optimal parameter setting for FSS-UCB, and then address the synchronization issues and the signaling for opportunistic channel sampling.

7.1 Parametrizing FSS-UCB algorithm

We start by discussing the frame size. In our current test-bed, the channel switching cost is 2-3ms, which is of order of a packet transmission time. In order to compensate for the switching cost, the AP mandates a frame size of 40 ms: approximately 10 packets of 1000B can be sent during one frame at the lowest transmission rate of 4.5 Mbps. Such a long frame size may cause short-term fairness issues. However, several papers, such as [35], report much shorter switching times, so we expect that 5-10 times shorter frame sizes will be enough to compensate the switching cost in more efficient radio designs.

To choose the rest of the parameters of the FSS-UCB algorithm, we use the tuned values derived from simulation experiments on measured channels: the optimal parameters are $\xi = 0.3$ and $\gamma = 1 - 5 \cdot 10^{-3}$. This implies $\gamma^{270} = 0.25$ and samples that are more than 10s away will contribute less than 0.25 to the quality estimates (because in our test-bed 270 frames last about 10s). This intuitively means that we provision for channel coherence times of a few tens of seconds[7]. We also use $\gamma' = \gamma$ and $q = 1/2$.

7.2 Synchronization issues

One of the main problems in any wireless system with channel hopping is to maintain synchronization: the receiver has to track the channel used by the transmitter to send packets. Any loss of synchronization will trigger a potentially costly re-discovery procedure. In the scenario where an AP implements FSS-UCB algorithm to transmit packets to several clients, it is important that all clients always remain synchronized with the AP. Indeed, clients need to overhear all packets sent by the AP to speed up the learning process (opportunistic channel sampling). We describe next a synchronization procedure in the AP scenario. The case of a single transmitter-receiver pair can be seen as a special case of this scenario.

Let i_t be the channel used by the AP to send packets in frame t. To inform all users of the channel that the AP will use in the next frame, i_{t+1} is encoded in the header of each

packet sent during frame t. To remain synchronized, a client u just need to successfully decode the header of one of the packets sent per frame. This happens with high probability. If the header of each packet of frame t cannot be decoded by client u, the latter will go to the *back-up* channel i_{t+1}^u such that $(i_{t+1}^u, l) = \arg\max_{(j,k)} w_{u,j,k}(t+1)$ and wait there. In other words, if client u looses synchronization, she will wait for the AP to transmit a packet on the channel that would be chosen for the next transmission destined to u. If synchronization cannot be recovered within a predefined time-out, client u starts a channel scanning procedure to track the AP.

Note that i_t^u can be locally computed by client u. But due to possible previous synchronization problems, client-u's estimate of i_t^u may be erroneous. In order for client u and the AP to have the same view of back-up channel, the AP includes in the header of packets sent to client u during frame t the value of i_{t+1}^u.

The above synchronization procedure requires the AP to announce what channel will be used in the next frame, and hence the execution of the FSS-UCB is delayed by one frame. Our experiments showed that this delay does not impact the performance of the algorithm.

7.3 Signaling for opportunistic channel sampling

Finally, we describe a low-overhead signaling procedure which allows clients to feed back the overheard channel estimates (Section 5.4) to the AP.

All clients are in sync with the AP, listen promiscuously and learn channel information from overheard packets. Each client u keeps a record $(o_{u,i,l}^R, \hat{x}_{u,i,l}^R, \hat{n}_{u,i,l}^R)$ for each channel i and rate r_l, where $o_{u,i,l}^R$ is the number of overheard packets and $\hat{x}_{u,i,l}^R$ and $\hat{n}_{u,i,l}^R$ are the discounted weights and number of samples, respectively. For each overheard packet on channel i, rate r_l, the client u increases $o_{u,i,l}^R$ and updates $\hat{x}_{u,i,l}^R$ and $\hat{n}_{u,i,l}^R$ using soft-samples and the updates (5) and (4).

How will the client classify an erroneous packet, given that its content is not reliable? As we observe in the measurements, it is much more likely that the packet payload will get corrupted, than the packet header. Hence, for most of the packets, successful or not, the client can extract the header information correctly and update the corresponding record. If this is not possible, the packet is simply ignored.

Whenever the client u sends an acknowledgment for a correctly received packet, it appends one of the records $(o_{u,i,l}^R, \hat{x}_{u,i,l}^R, \hat{n}_{u,i,l}^R)$ using the LRU policy, and erases the record by setting all variables to zeros. When the AP receives this update, it updates its own records using a cumulative discount

$$\hat{n}_{u,i,l}(t+1) = \gamma^{o_{u,i,l}^R} \hat{n}_{u,i,l}(t) + \hat{n}_{u,i,l}^R,$$
$$\hat{x}_{u,i,l}(t+1) = \gamma^{o_{u,i,l}^R} \hat{x}_{u,i,l}(t) + \hat{x}_{u,i,l}^R.$$

Note that this is *as if* the AP had received all the packet information itself, and performed the updates (5) and (4) accordingly.

---

[7]For faster implementations, with higher channel rates and smaller processing delays than ours, the minimum required coherence time can be significantly reduced.
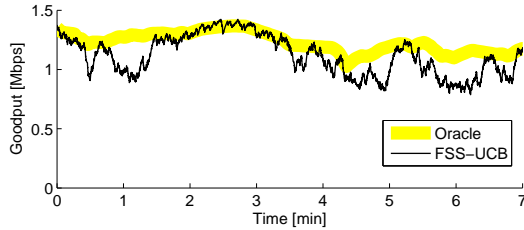
Figure 5: Goodputs achieved under FSS-UCB algorithm (solid black line) and the optimal Oracle algorithm always selecting the best (channel, rate) pair (thick yellow line) on link A-C.

8. PERFORMANCE EVALUATION

We evaluate FSS-UCB algorithm using both simulations and our SDR testbed.

8.1 Simulation Results

We first evaluate FSS-UCB using simulations for two reasons. First, we wish to compare FSS-UCB with an ideal or Oracle algorithm that always selects the best (channel, rate) pair, which is impossible to implement in a test-bed. Second, we wish to evaluate how the performance of FSS-UCB depends on the speed of channel changes (channel coherent time), which we can control in simulations but not in our testbed.

We use the channel traces from Section 3 to get realistic channel conditions. The transmission success of each packet is simulated as a Bernoulli random variable with the success probability corresponding to the success probability observed in the real traces, at that particular time instant, for the selected channel, rate and client. We compare FSS-UCB against two algorithms. The first one is an Oracle algorithm that knows the future success rates, and at every time instant selects the (channel, rate) pair with the highest success rate. The second is the random algorithm that randomly selects a (channel, rate) pair and uses that pair for the entire simulation.

We first compare FSS-UCB with the Oracle algorithm. We focus on a single link scenario for link A-C, and we plot the goodputs achieved by both algorithms in Figure 5. We observe that the performance loss of FSS-UCB is less than 5% compared to the Oracle! Thus FSS-UCB efficiently learns channel conditions, and is able to track the best (channel, rate) pair most of the time.

Next, we investigate how the performance of FSS-UCB depends on the speed of channel condition variations (channel coherence time). To do so, we again use the channel traces from Section 3, but we speed up the channel condition variations by a constant factor. We compare FSS-UCB with the Oracle and the random algorithms. In all cases we run the simulations over the entire trace 10 times and report the average rate achieved. We plot the ratio of the average goodputs achieved under FSS-UCB to the Oracle or to the random algorithm in Figure 6. We observe first that the random algorithm performs quite poorly. We also notice that as we speed up the channel variations, the performance of FSS-
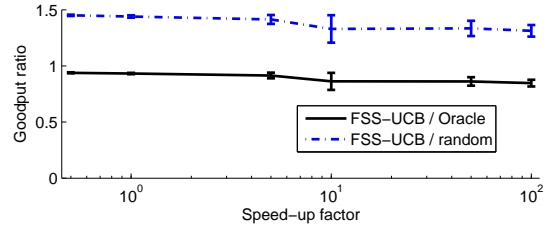

Figure 6: Ratio of the goodputs achieved by FSS-UCB algorithm and by the Oracle or the random algorithms as we speed-up channel variations.

UCB naturally drops. However FSS-UCB achieved 85% of the goodput of the Oracle algorithm even with a speed-up factor equal to 100, and it still performs 30% better than the random algorithm.

8.2 Experimental Results

We next evaluate the FSS-UCB algorithm in the testbed described in Section 3.1. We first evaluate FSS-UCB algorithm in a single-link scenario with link A-C and then in a 3-link scenario, both using live experiments.

We compare FSS-UCB algorithm to an algorithm that always selects the same channel, referred to as *single-channel algorithm* in the following. In the case of the single-channel algorithm, we *a priori* select one channel and run the whole experiment on that channel. To adapt the transmission rates with the single-channel algorithm, we use SampleRate [9], which in turns offers a performance similar to what we would have obtained by running our learning algorithm FSS-UCB on a single channel (just adapting the rates).

We generate synthetic traffic to always maintain the queues backlogged. In the case of the single-channel algorithm, and when multiple links are considered, we use a simple FIFO scheduler to define the order at which packets are sent by the AP. This contrasts with the FSS-UCB algorithm that targets some fairness guarantees.

Single-link Scenario: We first consider the single scenario, with link A-C. We run 6 experiments. The first experiment is with our learning algorithm FSS-UCB. The other five experiments are with the single-channel algorithm on channels 510, 530, 550, 580 and 600 MHz, respectively. We run each experiment for 30 min.

We present a snapshot of goodputs in Figure 7. Different channels exhibit different performance that varies over time, with a coherence time of order of seconds to minutes (an observation that is consistent with Figure 4). We also see that FSS-UCB manages to learn the best (channel, rate) well before the coherence time: it surfs on the top of the best channel, offering consistently good performance.

In Figure 8 we give the average goodput achieved in different experiments over periods of duration 30min. Our learning approach yields a 35.7% improvement over the average of other 5 experiments, 8.2% over the single-channel algorithm selecting the best channel and as much as 96.7% over that selecting the worst channel. This suggests that FSS-
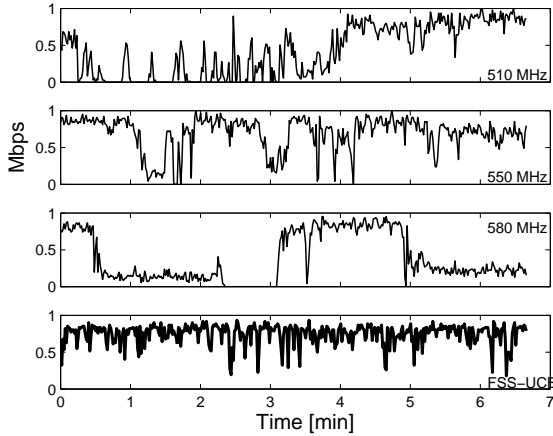
Figure 7: Goodputs vs. time of the single-channel algorithm on channels 510 MHz, 550 MHz and 580 MHz, and goodput vs. time of the learning algorithm FSS-UCB.

Channel [MHz]	510	530	550	580	600	FSS-UCB
Goodput [Mbps]	0.42	0.61	0.69	0.38	0.65	0.75
Improvement	76%	22%	8%	97%	15%	N/A

Figure 8: Average goodputs under the single-channel algorithm and under FSS-UCB over 30 min intervals and the relative improvements of FSS-UCB over the single-channel algorithm on different channels.

UCB is able to really track the best (channel, rate) pair.

Access Point Scenario: Finally, we look at the multiple client scenario. Three clients are served by the AP. Again we run two types of experiments, with the single-channel algorithm and with FSS-UCB. Running the single-channel algorithm on different channels yields similar goodputs. This is because the goodput is averaged over 30 min, a long period of time where all channels exhibit similar average quality. Thus we provide the results for 580MHz channel only. The results are given in Figure 9.

Link	A-B	A-C	A-D	Sum
Goodput [Mbps], Single channel	0.13	0.16	0.23	0.52
Goodput [Mbps], FSS-UCB	0.22	0.19	0.21	0.62
Improvement	63%	23%	-8.5%	19.2%

Figure 9: Average goodputs on different links of the single-channel algorithm on channel 580MHz and of FSS-UCB over 30min intervals, and the relative improvements of FSS-UCB over the single-channel algorithm.

Observe that FSS-UCB provides more balanced goodputs, i.e., improves fairness. The goodput of link A-D has decreased, but FSS-UCB also increased the previously smaller goodputs on links A-B and A-C. Moreover, in addition to the improve fairness, there is an overall goodput improvement of almost 20% when using FSS-UCB. We see that FSS-UCB works well with three links.

9. CONCLUSIONS AND FUTURE WORK

We have designed and implemented a learning algorithm, FSS-UCB, for dynamic rate and channel selection for wireless systems exploiting white spaces, and used both simulations and an SDR testbed to evaluate its performance. Using our algorithm, we observe an average performance gain of 40% over algorithms that always select the same channel. The gain can be as large as 100% when compared to the case where unfortunately the worst fixed channel is chosen for transmission. In a network setting where an AP serves multiple clients, our algorithm provides greater fairness and less variability than the single channel algorithm, and significantly increases the aggregate goodput. FSS-UCB is applicable to other similar environments, and we believe that its learning efficiency can be further improved by increasing PHY rates and decreasing different system delays – It should then be able to track the best channel and rate even when the channel coherence time is of the order of seconds.

The main limitation of our algorithm is that it has not been evaluated in a setting with multiple APs. We believe our algorithm will still work on the top of a CSMA-like MAC protocol, using techniques such as those described in [36] to distinguish between PHY and MAC layer collisions. Balancing the channel quality and medium access in an optimal and distributed way is still an open problem, as we discussed in Section 2, and left for future work.

10. REFERENCES

[1] FCC. Second memorandum opinion and order, FCC 10-174. http://www.fcc.gov/Daily_Releases/Daily_Business/2010/db0923/FCC-10-174A1.pdf, 2010.

[2] M. Mishra and A. Sahai. How much white space is there? Technical Report UCB/EECS-2009-3, UC Berkeley, 2009.

[3] J. Camp and E. Knightly. Modulation rate adaptation in urban and vehicular environments: cross-layer implementation and experimental evaluation. In *MOBICOM*, 2008.

[4] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *SIGCOMM*, 2010.

[5] S. Sen, N. Santhapuri, R. Choudhury, and S. Nelakuditi. CBAR: Constellation based rate adaptation in wireless networks. In *NSDI*, 2010.

[6] Ubiquiti. XR1 and XR7 radio modules. http://www.ubnt.com/xr1.

[7] Y. Yuan, P. Bahl, R. Chandra, P. Chou, J. I. Ferrell, T. Moscibroda, S. Narlanka, and Y. Wu. KNOWS: kognitiv networking over white spaces. In *DySpan*, 2007.

[8] H. Kim and K. G. Shin. In-band spectrum sensing in cognitive radio networks: Energy detection or feature detection? In *Proceedings of MOBICOM*, 2008.

[9] J. Bicket. Bit-rate selection in wireless networks. Master's thesis, MIT, 2005.

[10] M. Lacage, M. H. Manshaei, and T. Turletti. IEEE 802.11 rate adaptation: a practical approach. In *Proceeding of ACM MSWiM*, 2004.

[11] S. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *MOBICOM*, 2006.

[12] K. Liu, Q. Zhao, and B. Krishnamachari. Dynamic multichannel access with imperfect channel state detection. *IEEE Transactions on Signal Processing*, 58(5), 2010.

[13] Hai Jiang Lifeng Lai, Hesham El Gamal and H. Vincent Poor. Cognitive medium access: Exploration, exploitation and competition. *IEEE/ACM Trans. on Mobile Computing*, 10(2), 2011.

[14] Q. Zhao, L. Tong, A. Swami, and Chen Y. Decentralized cognitive MAC for opportunistic spectrum access in ad hoc networks: A POMDP framework. *IEEE Journal on Selected Areas in Communications*, 25(3):589–600, 2007.

[15] S. H. Ahmad, M. Liu, T. Javidi, Q. Zhao, and B. Krishnamachari. Optimality of myopic sensing in multi-channel opportunistic access. *IEEE Trans. on Information Theory*, 55(9):4040 – 4050, September 2009.

[16] J. Unnikrishnan and V. V. Veeravalli. Algorithms for dynamic spectrum access with learning for cognitive radio. *IEEE Transactions on Signal Processing*, 58(2):750 – 760, February 2010.

[17] J. Ai and A. Abouzeid. Opportunistic spectrum access based on a constrained multi-armed bandit formulation. *Journal of Communications and Networks*, 11, 2009.

[18] K. Liu and Q. Zhao. Channel probing for opportunistic access with multi-channel sensing. In *Asilomar*, 2008.

[19] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, 2002.

[20] A. Garivier and E. Moulines. On upper-confidence bound policies for non-stationary bandit problems. http://front.math.ucdavis.edu/0805.3415, 2008.

[21] P. Chaporkar and A. Proutiere. Optimal joint probing and transmission strategy for maximizing throughput in wireless systems. *IEEE Journal on Selected Areas in Communications*, 26(8):1546 – 1555, October 2008.

[22] S. Huang, X. Liu, and Z. Ding. Optimal sensing-transmission structure for dynamic spectrum access. In *IEEE INFOCOM*, 2009.

[23] S. Guha, K. Munagala, and S. Sarkar. Jointly optimal transmission and probing strategies for multichannel wireless systems. In *CISS*, 2006.

[24] N. Chang and M. Liu. Optimal channel probing and transmission for opportunistic spectrum access. In *MOBICOM*, 2007.

[25] Arquiva. http://www.arqiva.com/.

[26] D. Tse and P. Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[27] A. Sabharwal, A. Khoshnevis, and E. Knightly. Opportunistic spectral usage: bounds and a multi-band csma/ca protocol. *IEEE/ACM Trans. Netw.*, 15(3):533–545, 2007.

[28] P. Chaporkar, A. Proutiere, H. Asnani, and A. Karandikar. Scheduling with limited information in wireless systems. In *MobiHoc*, New York, NY, USA, 2009. ACM.

[29] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006.

[30] N. Ahmed and S. Keshav. SMARTA: a self-managing architecture for thin access points. In *CoNext*, 2006.

[31] A. Anandkumar, N. Michael, and A.K. Tang. Opportunistic spectrum access with multiple users: Learning under competition. In *IEEE Infocom*, 2010.

[32] A. Proutiere and G. Kasbekar. Decentralized opportunistic medium access in multi-channel wireless systems: A learning approach. In *Allerton*, 2010.

[33] Y. Gai, B. Krishnamachari, and R. Jain. Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation. In *DySPAN*, 2010.

[34] H. Kushner and G. Yin. *Stochastic approximation and recursive algorithms and applications*. Springer, 1997.

[35] F. Herzel, G. Fischer, and H. Gustat. An integrated cmos rf synthesizer for 802.11a wireless lan. *IEEE Journal of Solid-state Circuits*, 18(10), October 2003.

[36] J. Kim, S. Kim, S. Choi, , and D. Qiao. CARA: Collision-aware rate adaptation for IEEE 802.11 WLANs. In *Proceeding of IEEE Infocom*, 2006.