Cooperative Learning and Coordination for Cognitive Radio Networks

William Zame, Jie Xu, and Mihaela van der Schaar

Abstract

The *raison d'etre* for cognitive radio is that the radio spectrum is a scarce resource. Cognitive radio stretches this resource by enabling secondary users to operate in portions of the spectrum that are reserved for primary users but not currently used by the primary users. As it is whenever users share resources, coordination is a central issue in cognitive radio networks: absent coordination, there may be collision, congestion or interference, with concomitant loss of performance. Cognitive radio networks require coordination of secondary users with primary users (so that secondary users should not interfere with primary users) and of secondary users with each other. These coordination problems are especially severe because secondary users may frequently enter and exit channels within the network or the network itself. This paper proposes protocols that solve these coordination problems. These protocols are completely distributed (requiring neither central control nor the exchange of any control messages), fast (with speeds exceeding those of existing protocols), efficient (in terms of throughput and delay) and scalable. The protocols proposed rely on *cooperative learning*, exploiting the ability of users to learn from and condition on their own histories while simultaneously enabling the learning of other users. These protocols can be formulated as finite automata and hence implemented using currently existing technology. Analytic results and simulations illustrate the power of these protocols.

Index Terms Cognitive radio networks, cognitive medium access control, perfect coordination, cooperative learning in networks, distributed protocols

W. Zame: Distinguished Professor, Departments of Economics and Mathematics, UCLA, Los Angeles, CA, 90095

J. Xu: Department of Electrical Engineering, UCLA, Los Angeles, CA, 90095; Corresponding Author: jiexu@ucla.edu

M. van der Schaar: Professor, Department of Electrical Engineering, UCLA, Los Angeles, CA, 90095; Fellow, IEEE

Financial support was provided by National Science Foundation grants 0617027 and 0830556, by the UCLA Academic Senate Committee on Research, and by the Einaudi Institute for Economics and Finance (Rome). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any funding agency.

I. INTRODUCTION

Resources are scarce. This is no less true about the radio spectrum than it is about consumer goods – but substantial portions of the radio spectrum are wasted because they are "owned" by or "reserved" for users who need them for only a fraction of the time. Cognitive radio was introduced [1] to address this problem, allowing secondary users to access portions of the spectrum that are dedicated to primary users but not fully utilized. Following its introduction, a great deal of effort has been expended to improve the efficiency of cognitive radio networks. Much of this work has been dedicated to developing sophisticated physical layer technologies that allow secondary users to detect the presence of primary users and coexist with them in the same spectrum bands with minimal interference. Less attention has been paid to developing efficient MAC layer protocols that allow secondary users to coordinate with *each other*. (We discuss some of these literatures at the end of this Introduction.) Absent coordination, there may be collision, congestion or interference, with concomitant loss of performance. Coordination problems are important even in the absence of primary users (as would be the case in cognitive radio networks operating in unlicensed bands such as 2.4GHz and 5GHz), and become even more important in the presence of primary users, because secondary users may frequently enter and exit channels within the network or even the network itself. Efficient methods of coordination have the potential to avoid conflict, promote sharing, and improve utilization of network resources, increasing throughput and decreasing delay. Such coordination can be obtained if the network is centrally controlled or if the users can exchange control messages – but central control and the exchange of control messages between users (or stations) are typically wasteful of resources and often impossible. This means that *distributed* protocols are necessary.

In this paper, we propose protocols that solve these coordination problem(s). These proposals are *completely distributed* (requiring neither central control nor the exchange of any control messages), *fast* (with speeds exceeding those of existing protocols), *efficient* (in terms of throughput and delay) and *scalable* (adapting easily to both small and large numbers of users).

The key to our protocols is that they lead users to learn about the evolving state of the system and condition their pattern of actions on what has been learned. In our protocols, users learn more and use more of what they have learned than in other protocols: users learn, not only about the channel, but also about the pattern of actions of *other* users, and they condition their current

actions not only on their current observations but also on their past histories (more precisely, on a summary of past histories). Our protocols require that users learn from their own observations and also choose actions that promote the learning of other users: i.e., users learn *cooperatively*. To this end, our protocols exploit two opposite facets of the interaction: on the one hand, that actions (transmissions) convey information, and on the other hand, that the absence of actions (silence) can also convey information. Embedded within our initialization protocol is the idea that users that are indistinguishable *ex ante* and randomize in an identical fashion (but independently) may experience different realizations of their randomization and hence become distinguishable *ex post*. We believe this paper is the first to exploit the combined power of these forces.

The protocols we propose require only finite memory and can be formulated as finite automata; they can be implemented using current technology, and do not require development or deployment of any new hardware. (See [2] for instance.)

We propose a number of protocols, serving a number of purposes. The *Initialization Protocol* enables initial coordination among a group of *ex ante* identical secondary users. The *Steady State Protocol* enables an initialized group of secondary users to access the channel in sequence, with no collisions and no idle slots or a single idle slot. We propose two basic *Entry Protocols* and a variant of each. The first of the basic entry protocols enables a coordinated group of secondary users to accommodate an entrant who is known to be a secondary user; the second enables a coordinated group of secondary users to recognize whether an entrant is a primary user, to whom the secondary users must defer, or another secondary user, whom they will accommodate. The basic entry protocols leave no idle slots; the variants of these protocols deliberately leave an idle slot (in parallel with the Steady State protocol). The reason for introducing serveral different protocols is that each may be more suitable in a particular environment. The choice between basic entry protocols is dictated by whether or not secondary users can physically distinguish primary users.[1] The choice to leave an idle slot or not is dictated by the specific characteristics of the network, especially the number of secondary users and the rate at which secondary users hop between channels; for various values of these parameters. As we discuss in Section III and illustrate by simulations in Section IV, idle slots may actually promote efficiency by promoting coordination across channels.

[1]This ability is frequently assumed in the literature [30], but it is not universally true of all cognitive radio networks.

A crucial issue with respect to coordination in our setting and in many others is the capacity of users to observe the state of the network and especially the actions of other users. In this paper we assume that users who attempt to access the network in a given period/slot can observe whether or not they were successful and hence whether or not other users were attempting to use the resource in the same period/slot (typically because they receive acknowledgement of transmissions), but that users who do not attempt to use the resource in a given period/slot can observe only whether or not the resource is in use – e.g., whether or not the channel is busy – but cannot differentiate between the situation in which a single other user has successfully accessed the resource and the situation in which several other users have attempted to access the resource and collided. This seems to us to be the most realistic informational assumption in the context of cognitive radio, but other assumptions have also been made in the literature, and might be sensible in other contexts. For instance, [3] assumes that users who do not attempt to use the resource can observe nothing at all, while [4] assumes that all users - those who attempt to access the resource and those who do not – can observe and differentiate between no attempt to access (an idle channel), successful access (a busy channel), and an unsuccessful attempt to access (a noisy channel). For further discussion of these different assumptions, see Section II.

We emphasize that our protocols prescribe behavior – actions conditional on histories – for *individual* users. This is of course necessary because users are indistinguishable to the designer (hence must be given the same instructions) and because users observe only their own histories and not the histories of other users. However, as we have pointed out above, users that follow the same instructions but experience different realization of their own randomizations need not experience the same histories nor take the same actions.

Some of the ideas exploited in this paper – randomization, collisions as a coordination device, and binary search - are perhaps familiar from the literature. (For instance, see the literature on slotted Aloha protocols [5][6][7] and on exponential backoff protocols [8][9].) Others – silence as a message, cooperative learning, conditioning current actions on past history – are less familiar. (see [3] for an example) The most novel single idea in this paper is perhaps the systematic reliance on both actions (attempts to access/transmit) and inactions (silences) as implicit messages. However, the overall novelty is not so much in any single idea as it is in exploiting the power available by combining *all of them*.

The introduction of cognitive radio networks [1] has prompted a large literature, which has

two main parts. The first part is dedicated to developing sophisticated physical layer technologies that enable secondary users to detect the presence of primary users and coexist with the primary users without interfering with them by transmitting only within identified spectrum "holes" – spectrum opportunities in which primary users are not active. For instance, IEEE 802.22, which has emerged as the standard for Wireless Regional Area Networks, allows secondary users to broadcast in "white spaces" – utilizing the geographically unused spectrum in the frequency bands allocated to television broadcast – and operates on a non-interfering basis to bring broadband access to hard-to-reach, low population density areas. In IEEE 802.22, co-existence with primary users is achieved by detecting the presence of primary users using information about the geolocation in which these users transmit, using beacons, or sophisticated spectrum sensing technology [10][11][12]. (A number of spectrum sensing technologies have been proposed for this purpose: energy detection based sensing [13][14], waveform-based sensing[15][16], cyclostationarity-based sensing [13][17], and radio identification based sensing [18].)

The second part of this literature develops MAC protocols for cognitive radio networks; for surveys see [26][27][28]. Within this literature there are two distinct strands of research, addressing two distinct problems. The first strand of research focuses on how secondary users can opportunistically identify the vacant portions of the spectrum and transmit in them while ensuring that the primary users are minimally affected; representative work on this strand includes [19][20][21][22]. The second strand of research aims at developing protocols that enable secondary users operating with the secondary network to coordinate with each other when accessing the spectrum opportunities that have been identified. A key challenge in this regard is how users can/should optimally adapt their transmission strategies (including hopping among various frequency bands). Examples of work on this strand include [23] (which designs efficient auction mechanisms), [25] (which offers an alternative approach based on the famous Vickrey-Clark-Groves mechanism) and [24] (in which secondary users to compete with each other for the limited and time-varying spectrum opportunities provided by a central spectrum moderator). However, all of this work relies on a centralized coordinator to allocate the spectrum opportunities among the secondary users. These methods are thus inefficient because they are costly, and require much information exchange. Closer to our work is [29], which proposed a class of distributed MAC protocols with memory intended to achieve efficient spectrum sharing among

secondary users while protecting the primary user from potential interference by the secondary users. As do we, [29] allows for the possibility that secondary users cannot physically distinguish primary users from secondary users and that coordination messages cannot be exchanged between a user and a central controller or between users, suggesting instead a role for observed patterns in history as a substitute for message exchange. However, that work considers only a simple protocol based on users past transmission histories, and this protocol does not guarantee that perfect coordination among secondary users emerges either quickly or with high probability, and so fails in utilize the available spectrum capacity. By contrast, our protocols are fully distributed, require neither information exchange nor a dedicated control channel, attain perfect coordination very quickly and with very high probability – and thus utilize all (or almost all) of the available spectrum capacity.

Following this Introduction, Section II formalizes the environment, the informational structure and the nature of the protocols we use. Section III presents and analyzes our proposed initialization protocol; Section IV presents and analyzes our proposed re-coordination protocols. Section V presents simulations and comparisons with existing protocols. Section VI concludes.

II. FRAMEWORK

We consider the interaction among N users over a (potentially) infinite time horizon. Time is divided into discrete *slots* or *periods* indexed by $t = 0, 1, \ldots$. At each time, each user must choose whether to transmit or not (remain silent); we use $1, 0$ (respectively) for these decisions. We write $A = \{0, 1\}$ for the set of possible actions of each user, $A = \{0, 1\}^N$ for the set of possible action profiles and $a^t = (a_1^t, \ldots, a_N^t)$ for the vector of actions of the N users in slot t. The set of *channel states* is $S = \{s_0, s_1, s_2, \ldots s_N\}$. Given the vector a^t of transmissions at time t the resulting channel state is

$$s^t = s_m \iff \sum a_i^t = m \tag{1}$$

The channel can support exactly one transmission at a given time: if more than one user transmits, the result is a collision and no transmissions are successful. Thus, we refer to state 0 as *idle* and state 1 as *success* and to the remaining states as *collision*.

A. Information

A number of different assumptions about what users can observed are considered in the literature. In our protocols we will make specific and particular assumptions about what users can observe, but we allow for other possibilities in order to facilitate comparison with other work. We formalize the information available to a user in terms of *information partitions* of the set S of channel states. Throughout, we assume that no user can ever distinguish between attempted transmission by m users and n users when $m, n \geq 2$ (so that there is a collision). Hence the information partition of a user is a partition of S whose elements belong to the family of subsets

$$\{s_0\}, \{s_1\}, \{s_0, s_1\}, \{s_0, s_2, \ldots s_N\}, \{s_1, s_2, \ldots s_N\}, S$$

We consider three particular information conditions; our focus here is on the second; we present the other two for purposes of comparison. In each case we specify the information partition of a user that transmits (attempts to access the resource) and a user that is silent (does not attempt to access the resource).

(i) *No Sensing:*

- **Transmit** $\{s_0\}, \{s_1\}, \{s_2, \ldots s_N\}$
- **Silent** S

(ii) *Sensing:*

- **Transmit** $\{s_0\}, \{s_1\}, \{s_2, \ldots s_N\}$
- **Silent** $\{s_0\}, \{s_1, \ldots s_N\}$

(iii) *Sensing + Collision-Detection:*

- **Transmit** $\{s_0\}, \{s_1\}, \{s_2, \ldots s_N\}$
- **Silent** $\{s_0\}, \{s_1\}, \{s_2, \ldots s_N\}$

In words: in all information conditions, users that transmit observe whether the channel is idle (which is impossible, given that they transmit) or there is a successful transmission or there is a collision. In the No Sensing condition, users that do not transmit observe nothing. In the Sensing condition, users that do not transmit observe whether the channel is idle or busy – but not whether there is a successful transmission or a collision. In the Sensing + Collision-Detection condition, all users observe whether the channel is idle or there is a successful transmission or there is a collision. (Here and in various other places, we use the words idle, success, collision

and busy rather than specifying the corresponding set in the appropriate partition; this should cause no confusion.)

These different informational conditions reflect very different settings and support very different possible behaviors because users can only condition on what they can observe. On the one hand, if all users can observe whether the channel is idle or there is a successful transmission or there is a collision – as assumed by [4] – then all users can condition on these various events. However, if – as seems most realistic – users who do not transmit can observe whether or not a channel is busy but cannot observe the *difference* between a successful transmission and a collision, they cannot *condition on* the difference between a successful transmission and a collision. As we shall see in our Initialization Protocol, it is important at various stages for users to learn precisely this; if they cannot learn this *directly* – from their own observations – they must learn it *indirectly* –, from the subsequent actions of other agents. To enable this indirect learning, users must learn *cooperatively*. On the other hand, if users who do not transmit observe nothing – as assumed by [3] – even indirect learning will be enormously slower and more difficult, and there will be some things that users will simply *never* be able to learn. As we have commented, the informational condition assumed in this paper seems to be the most realistic description of most environments – most especially, congnitive radio environments.

B. Protocols

At each time, each user must choose an action 0 (remain silent) or 1 (transmit); a *protocol* is a set of instructions for making such choices. We allow choices to be random and to depend on the personal history of the user. To make this precise, it is convenient to define *channel events* and *channel histories* and *personal events* and *personal histories*.

- A *channel event* is an element of $(A^N \times S)$; a *channel history of length* $T \geq 0$ is an element of $(A^N \times S)^T$; *an infinite channel history* is an element of $(A^N \times S)^\infty$. A channel event specifies the actions taken by all users and the resulting channel state at a given time; a channel history of length T specifies the actions taken and the resulting channel state at each time $t = 0, 1, \ldots T$; an infinite channel history specifies the actions taken and the resulting channel state at each time $t = 0, 1, \ldots$.[2] We write $\mathcal{H}_c(T)$ for the set of channel

[2]Since the channel state depends deterministically on the actions, it is redundant to specify both the vector of actions and the channel state, but the redundancy is convenient.

histories of length T, $\mathcal{H}_c = \bigcup \mathcal{H}_c(T)$ for the set of channel histories of finite length, and $\mathcal{H}_c(\infty)$ for the set of infinite channel histories.

- To simplify notation, we label each set in the information partition of a user by its first state. With this notation we define a *personal event* to be an element of $(\{0,1\} \times S)$ and a *personal history* of length T to be an element of $(\{0,1\} \times S)^T$. (Note that s_1 is the label for the set $\{s_1\}$ in the information partition of a user that transmits and is also the label of the set $\{s_1, \ldots, s_N\}$ in the information partition of a user that does not transmit. However, because a personal event includes the action of the user, no confusion should arise.) A personal event specifies the action taken by a particular user and the observation made by the user. We write $\mathcal{H}_p(T)$ for the set of personal histories of length T, $\mathcal{H}_p = \bigcup \mathcal{H}_p(T)$ for the set of personal histories of finite length and $\mathcal{H}_{p_*}$ for the set of infinite personal histories.

In both cases, note that the empty history is the unique history of length 0. A *protocol* is a function $f : \mathcal{H}_p \to \Delta(\{0,1\})$.[3] That is, a protocol specifies a (random) choice of action at each time, conditional on what was observed by the user at previous times: the date, the action of the user and the information obtained about channel states.[4]

A protocol is a set of instructions for *each user*. Because users are *ex ante* identical, we insist that the *same* protocol be specified for each user. However, we assume that *users randomize independently*, so *ex post*, different users may experience different realizations, choose different actions and experience different histories, even though they are following the same protocol.

III. The Initialization Protocol

In this section we describe the proposed initialization protocol and establish an analytic estimate for the speed of convergence; for simulations see Section V.

Throughout we assume there are N users - but we emphasize that N need not be known. (As noted earlier, in order that our protocol be implementable by a finite automaton, it is necessary that *some* upper bound on the number of users be known, but the specific upper bound does not enter either the protocol or our estimate of the speed of convergence.) We write Z for the set of users and $z \in Z$ for a generic user. The protocol is built around a binary sort. Of course, a

[3]We write $\Delta(\{0,1\})$ for the set of probability distributions on $\{0,1\}$.

[4]We allow for protocols that condition on the previous actions of the user but our protocols do not do so.

TABLE I

CODES, NAMES AND CARDINALITIES OF THE USER SETS

Code	Name	Cardinalities of the Sets
11	HIT	$\|Z_{RT}\| \geq 1, \|Z_{RS}\| \geq 1$
0	IDLE	$\|Z_T\| = \|Z_{RT}\| = 0$
100	NOISE	$\|Z_{RT}\| \geq 2, \|Z_{RS}\| = 0$
101	WIN	$\|Z_T\| = 1, \|Z_{RT}\| = \|Z_{RS}\| = 0$

binary sort can be easily and efficiently implemented by a central authority; because we have no central authority, our binary sort must be executed in a completely decentralized fashion. This requires each user to *learn* and to *enable the learning of other users*.

By definition, a protocol is a set of instructions for each individual user. At each step/slot of the protocol, the instruction will be to transmit, remain silent, or randomize (depending on the user's personal history). For simplicity, our protocol only uses randomization that puts equal weight on transmission and silence. With the obvious notation, we write $T, S, R = .5T + .5S$ for transmission, silence, randomization (respectively). It is important to keep in mind the distinction between the instruction to transmit or remain silent and the action taken following the realization of randomization.

It is useful to introduce some notation and terminology. Consider the actions of users in a single slot. Write Z_T, Z_S, Z_R for the set of users whose instruction is to transmit, remain silent, randomize (respectively) in this slot, and Z_{RT}, Z_{RS} for the sets of users who are randomizing in this slot and whose realizations are transmit T and remain silent S respectively. Each user knows to which of these sets it belongs, although it may know little or nothing about the other sets, even whether they are empty or not empty; much of our protocol is designed precisely to make these facts common knowledge. We identify four possibilities for what actually occurs in this slot, distinguished by the cardinalities of the sets Z_T, Z_{RT}, Z_{RS}. For later convenience, we give each of these possibilities a binary code (which represents what will occur in the current slot and perhaps in the following slot(s)), and a name, presented in Table I (This table does not exhaust all the possibilities that might occur but it does exhaust the possibilities that might occur when users follow the protocol.). The names we have adopted are intended to be suggestive: A binary sort successively partitions users into two groups until the remaining sets are all singletons. In our protocol, the partitioning is carried out endogenously by having users randomize. HIT means that

the current set of users is indeed partitioned – i.e., not all users who were randomizing received the same realization; IDLE means that no user transmitted and in particular that all users who were randomizing received the realization S; NOISE means that that at least two users were randomizing and all users who were randomizing received the realization T; WIN means that a singleton has been reached. (As we will see, this can only occur when the singleton user is *not* randomizing.) From the point of view of randomized sorting, HIT and WIN represent successes, IDLE and NOISE represent failures; we will eventually need to count successes and failures.

We emphasize that *no single user needs to know (in the current slot) which of these has occurred*; indeed, for HIT, NOISE no single user *will know* (in the current slot) which of these has occurred. Our protocol will lead each user to *learn* which of these has occurred – but that process will require a specific set of actions over the succeeding two slots. Because the protocol must be completely distributed, the action of any user in any slot must depend only on (a summary of) the user's history up to that slot. To be sure that this is the case will require some care. The formal description of the protocol is fussy so we relegate it to the Appendix. Below we give a less formal – but correct – description.

We describe the first cycle of the protocol; succeeding cycles are almost – but not exactly – the same. In the first slot all users randomize $R = .5S + .5T$; users who receive the realization S are silent, users who receive the realization T transmit. If all users are silent, the channel will be IDLE and all users begin the cycle again. If at least one user transmits, then all users wait for the next slot. In the next slot, users who were silent in the first slot transmit and users who transmitted in the first slot are silent. If at least one user transmits in this slot then there is a HIT and the set of users has been partitioned into two non-empty branches. It is important that the users in each branch know if they are alone in their branch – and they do: they are alone in their branch if and only if their transmission (in the first or second slot of the cycle as the case may be) was successful. If the user that transmitted in the first slot was alone in its branch then it will follow the pattern T, S, T in the next cycle (and the protocol will insure that other users will be silent); if the user that transmitted in the second slot was alone in its branch then it will be silent in the next cycle and follow the pattern T, S, T in a later cycle (when its turn arises). If no user transmitted in the second slot then it must be that all users transmitted in the first slot, so we have NOISE, and all users begin the cycle again.

Cycles after the first cycle differ in several ways. If the cycle realization is HIT then users

whose realization in the first slot was S become inactive and remain silent for some time. If the cycle realization is HIT and some user (or users) is alone in a branch, later cycle (or cycles) will follow the unique WIN pattern (no randomization), all users can recognize this and so can count the number of times it has happened and when it (eventually) happens to them – the latter is their *index*. When this happens, users remember these numbers and retire (remain silent) until the protocol stops, at which point the users will be completely ordered from 1 to N by their indices. (At this point users will also discover the value of N, the number of users.) Every time the cycle outcome is WIN, the remaining users adjust counters that inform each user when its turn to become active again arises, and all users begin the cycle again.

A. Convergence

The fundamental facts about the Initialization Protocol that it converges (with probability 1) and converges quickly (with high probability). The speed of convergence can be expressed in a number of ways. Perhaps the simplest and easiest to understand is estimate the number of slots that must elapse in order that the probability of convergence exceeds some pre-specified number (less than 1). The following theorem provides the estimate we desire and also the guarantee that the Initialization protocol converges with probability 1; we defer the proof to the Appendix (following the more formal algorithmic description of the protocol).

Theorem Fix $D > 0$. If $K > 7N + 3D + 12(ND + D^2/4)^{1/2}$ then the probability that the Initialization Protocol converges in no more than K slots is at least $1 - 2e^{-D}$.

In a typical cognitive radio network the number of secondary users would typically be on the order of $N = 10$ or $N = 20$. For instance, if $N = 10$ our estimate shows that convergence with probability at least 0.998 requires fewer than 240 slots; if $N = 20$ our estimate shows that convergence with probability at least 0.998 requires fewer than 370 slots. In fact, as the simulations in Section V demonstrate, these estimates are very conservative, because the probability that early cycles end in HIT is much greater than $1/2$, a fact of which we have not taken account but which makes for many fewer "misses".

IV. Exit and Entry

The initialization protocol coordinates secondary users. If secondary users always entered the network and exited (a channel or the network) at the same time, the Initialization Protocol is

all that would be required. However this is seldom or never the case: secondary users come and go at need and exit temporarily or permanently or hop to a different channel when a primary user (who has rights to the channel) arrives. Depending on the network, these events may occur very frequently, so it is important that they be handled quickly and smoothly. Here we propose protocols that accomplish that accomplish purpose. We note here that these "re-coordination protocols" are typically *much* faster than the Initialization Protocol so it is much more efficient to employ these re-coordination protocols when possible, rather than to rely on the Initialization Protocol whenever the order of transmission is "disturbed" in some way.

Our entry protocols take cognizance of the fact that there are a number of different scenarios of interest in cognitive radio deployments, and a protocol that is appropriate for single-channel networks may not be appropriate for multi-channel networks, and a protocol that is appropriate when primary users can be recognized by their signatures may not be appropriate when primary users can not be recognized by their signatures. (The former situation is the one most treated in the literature [30][20] but both are realistic.) Moreover, as we shall show in the simulations, the efficiency of the various entry protocols depends not only on parameters of the network, including the number of channels and the requirements of the primary users, but also on how we measure efficiency, in particular on the weight given to goodput versus delay.

In what follows we assume entry and exit do not occur too often; more precisely, we assume that at most one entry or exit occur during these protocols. Since the number of slots required for any of these protocols to converge is at most a small multiple of the number of current secondary users (in a given channel), this seems a reasonable assumption.

A. Steady State Protocols

Before proposing exit and entry protocols we settle on the obvious steady-state protocol. After an initialization stage, each of the N users knows the total number of users and the user's own place in the ordering of these N users. For reasons that will become clear below, we allow for the possibility that the steady state protocol calls for K idle slots, where $K = 0$ or 1. Hence we assume (1) all users know the number N of users; (2) all users know the number K of slots that are to be left idle after each cycle of transmissions; (3) all users know the order in which they should transmit; (4) all users transmit in turn.

Steady State Protocol Immediately after the initialization protocol is complete, each of the N users transmits in the slot corresponding to its index $w(z)$. After the initial transmission, each user transmits every $N + K$ slots.

Note that each round of the Steady State Protocol consists of successful transmissions in N consecutive slots followed by $K = 0$ or 1 idle slots.

B. Exit Protocol

We assume that there are N current users and that they are following the Steady State Protocol above, with K idle slots. The exit protocol is very simple:

Exit Protocol When a user exits, the remaining users see an idle slot when they expected to see a busy slot, so they know there is one less user. They set $N^* = N - 1$ and continue to follow the Steady State Protocol with $N^* = N - 1$ in place of N; i.e. wait $N^* + K = N - 1 + K$ slots before transmitting again.

Following an exit, the remaining users maintain perfect coordination, transmitting in turn for $N^* = N - 1$ slots and then leaving K idle slots.

C. Entry Protocols

Our entry protocols must address two polar opposite circumstances, according to whether or not Primary Users can be physically distinguished from Secondary Users. If Primary Users can be physically distinguished from Secondary Users, we assume (as does the literature) that this is immediately recognized by all Secondary Users, who simply suspend transmission until the Primary User leaves and then resume where they left off. (If some Secondary Users have hopped to other channels in the meantime, the Exit Protocol will eliminate excess idle slots.) In this circumstance, we need only be concerned with entry of Secondary Users. If Primary Users can not be physically distinguished from Secondary Users, Primary Users are still entitled to exclusive use of the channel when needed, so the entry protocol itself must provide a means to distinguish between Primary Users and Secondary Users. This is easily accomplished via a small change in the protocol.

Before describing the several entry protocols it is useful to make some preliminary remarks. Suppose, for the sake of concreteness, that N secondary users are currently operating under the

Steady State Protocol with no idle slots. In order for an additional user to enter and the resulting $N+1$ users to again become perfectly coordinated, three things must happen: the Current Users must learn that there is an Entrant, the Current Users must "make room" for the Entrant, and the Entrant must learn the number of users. In order for the Current users to learn that there is an Entrant, something must happen that they do not expect, and the only thing this could be is a collision. Because users who do not transmit cannot distinguish between a successful transmission and a collision, *each* of the Secondary Users must experience a collision. (Having experienced a collision, the Secondary Users can "make room" simply by waiting one additional slot.) For its part, the Entrant can only learn if something happens that *it* does not expect, and the only thing this could be is a success. But one success is not enough – it must experience two successes, so that it learns the number of secondary users as the length of the interval between these two successes. The Entry Protocol SU-0 (entry protocol for a secondary user with 0 idle slots) simply arranges these events in a convenient way. If the Entrant is a Primary User, but cannot be physically distinguished from a Secondary User, the protocol can be "tweaked" to distinguish a Primary User Entrant from a Secondary User Entrant by giving the Entrant an opportunity to transmit (if it is a Primary User) or not transmit (if it is a Secondary User) in a slot in which those actions are guaranteed to be recognized *because the Current Users have left the slot (temporarily) idle*. Again, the Entry Protocol U-0 (entry protocol for an unknown user with 0 empty slots simply arranges these events in a convenient way.

At this point the reader may already see why using a Steady State Protocol that leaves an idle slot might be an advantage: it makes it possible for something else to happen that the Current Users do not expect, namely a busy slot where they expected to see an idle slot. Once again, the Entry Protocols SU-1 and U-1 arrange the appropriate events in a convenient way.

We first describe the the Entry Protocols for Secondary Users assuming that primary users can be physically distinguished (so that the protocol does not need to distinguish them). As before, we assume the current users are following the Steady State Protocol. The current users know K and N; the new/entering user knows K (it is a parameter of the system) but does not not know N. The protocols for $K = 0$ and $K = 1$ are quite different because all users can distinguish a busy channel from an idle channel but only users who are transmitting can distinguish a successful transmission from a collision.

Entry Protocol SU-0 The Entrant transmits in *every slot* until it experiences one successful transmission, then one collision, then one more successful transmission, in that order – but not necessarily consecutively (e.g., SSSCCCCS). At this point the Entrant sets N^* equal to the number of successive collisions plus 1 (this will be the total number of users including the Entrant.) From that point on, the Entrant follows the Steady State Protocol and transmits every N^*+K slots. Each Current User follows the Steady State Protocol until experiencing a collision. Collision reveals that a new user (necessarily a Secondary User in this case) has appeared so the Current User "makes room" by setting $N^* = N+1$ and follows the Steady State Protocol with N^* in place of N; i.e., transmit every N^*+K slots.

It is easily seen that after $2N-1+K$ slots the entrant and current users will agree that the new number of users is $N^* = N+1$ and will be perfectly coordinated, transmitting in turn for N^* consecutive slots, leaving K slots idle, then beginning again.

Entry Protocol SU-1 The Entrant waits until it observes 2 idle slots. The number of busy slots between successive idle slots is the current number of users N, so the Entrant sets $N^* = N+1$ and the Entrant follows the Steady State Protocol, transmitting every N^*+K slots. Each Current User follows the Steady State Protocol N, K until it observes a busy slot where it expects an idle slot; it then knows a new user (necessarily a Secondary User in this case) has appeared so the Current User "makes room" by setting $N^* = N+1$ and follows the Steady State Protocol with N^* in place of K; i.e., transmits every N^*+K slots.

If Primary Users cannot be distinguished physically from Secondary Users, we modify the Entry Protocols abovel so that the protocol itself makes it possible for Secondary Users to distinguish between Primary users and Secondary users; to do so we make the current Secondary Users wait longer/make a little extra room (temporarily) so that a Primary User can make itself known. If the entrant is revealed to be a Primary User, the current users wait until the Primary User has departed and then resume from where they left off. If the entrant is revealed to be a Secondary User, the current users and the entrant run the Exit Protocol (to eliminate the extra waiting slot that was created); at that point all users are perfectly coordinated, transmitting successively for $N^* = N+1$ slots and leaving K slots idle.

Entry Protocol U-0 A Primary Entrant transmits in every slot until it has had as many successful transmissions as it requires; then it leaves. A Secondary Entrant transmits until it experiences one successful transmission, then one collision, then one more successful transmission, in that order, but not necessarily consecutively (e.g., SSSCCCCS). Then it waits (remains silent) for $N+K+2$ slots, then transmits again. From that point on, it sets $N^* = N+1$ and follows the Steady State Protocol N^*, K, transmitting every N^*+K slots. A Current User follows the Steady State Protocol N, K until it experiences a collision. At that point it knows there is an entrant but does not know whether the Entrant is a Primary User or a Secondary User, so it waits $N+K+2$ slots. If no idle slot has been observed in that time, the Entrant is a Primary User, so the Current User waits until an idle slot has been observed and returns to the Operating Protocol with N, K as before; if an idle slot has been observed, the Entrant is a Secondary User so the Current User follows the Steady State Protocol with N^* in place of N, transmitting every N^*+K slots.

As noted, if the entrant is a Primary User then the current Secondary Users will not observe an empty slot until the Primary User has left, at which point they will simply resume the Operating Protocol. If the entrant is a Secondary User then the current Secondary Users will observe an empty slot at a particular place, hence know that the entrant is a Secondary User, and re-adjust as before.

Entry Protocol U-1 The Primary Entrant transmits every slot until it has had as many successful transmissions as it requries; then it leaves. A Secondary User waits until it has seen 2 empty slots; the number of busy slots between successive idle slots is the current number of users N, so the Entrant sets $N^* = N+1$, waits N^*+2 slots and transmits, then follows the Steady State Protocol N^*, K, transmitting every N^*+K slots. Each Current User follows the Steady State Protocol N, K until it observes a busy slot where it expects an idle slot; it then knows there is an entrant but does not know whether the entrant is a Primary User or a Secondary User, so it waits $N+K+2$ slots. If no idle slot has been observed in that time, the Entrant is a Primary User, so the Current User waits until one more idle slot has been observed and returns to the Operating Protocol with N, K as before; if an idle slot has been observed in that time,

the Entrant is a Secondary User so the Current User follows the Steady State Protocol with N^* in place of N, transmitting every $N^* + K$ slots.

As noted, if the entrant is a Primary User then the current Secondary Users will not observe an empty slot until the Primary User has left, at which point they will simply resume the Operating Protocol. If the entrant is a Secondary User then the current Secondary Users will observe an empty slot at a particular place, hence know that the entrant is a Secondary User, and re-adjust as before.

The designer should choose the Entry Protocol that is compatible with the technology (whether a Primary User can be detected physically) and with the Steady State Protocol (in terms of number of empty slots).[5] We will assume that this is done, and refer to the combination of Steady State Protocol and Entry Protocol as the *Operating Protocol*.

D. A Role for Idle Slots

We have allowed for the use of Operating Protocols that leave idle slots because although idle slots create ineffiency within the group of secondary users operating in a single channel they may also promote efficiency across channels. When primary users enter, secondary users are "bumped" and hence have an incentive to "channel hop". If Primary Users can be detected physicially, then using an Operating Protocol that leaves an empty slot saves many collisions and hence improves goodput. Whether or not Primary Users can be detected physically, using an Operating Protocol that leaves an empty slot makes it possible for a potential Secondary User entrant to know how many users are currently in each channel, and hence to hop to a channel with fewer users. In the long run, this does not matter for goodput but it does matter for delay, since delay is minimized by spreading users evenly across channels.

V. SIMULATIONS

A. Convergence Speed of the Initialization Protocol

Our first set of simulations investigates the speed of convergence of our Initialization Protocol (IP) – that is, the number of slots required for the users to achieve perfect coordination with some pre-specified probability. Theorem 2 provides an estimate of the speed of convergence.

[5]Notice that the designer specifies K but not N.

D		3	4	5	6	7	8
Probability		0.9004	0.9634	0.9865	0.9950	0.9982	0.9993
N = 10	Sim.	65	70	74	78	82	87
	Est.	173	191	208	224	239	253
N = 20	Sim.	127	133	139	144	149	153
	Est.	281	305	327	347	366	383
N = 50	Sim.	310	319	326	333	340	346
	Est.	467	603	635	664	691	716
N = 100	Sim.	612	624	634	643	651	659
	Est.	1003	1052	1095	1135	1172	1206

TABLE II

ESTIMATES AND SIMULATION RESULTS OF CONVERGENCE SPEED.

Probability		0.9	0.95	0.99	0.999
N=10	IP	65	68	75	84
	L-ZC	77	89	118	165
N=20	IP	127	131	140	151
	L-ZC	178	204	263	348
N=50	IP	310	316	329	343
	L-ZC	531	595	743	955
N=100	IP	612	620	636	658
	L-ZC	1191	1326	1618	2075

TABLE III

CONVERGENCE SPEED COMPARISON.

However, as we pointed out, this estimate is conservative; as the simulations show, convergence is actually much faster. Table II shows the estimates and the simulated number of slots for the corresponding probabilities for $D = 3, \ldots, 8$.

Our Initialization Protocol is faster than existing methods. For illustration, Table III compares (simulated) speed of convergence of our Initialization Protocol with the L-ZC (Zero Collision with Learning) protocol proposed in [31], which is an improvement on the ZC (Zero Collision) protocol propsed in [32]. The simulation shows that L-ZC requires much more slots to achieve a high convergence probability. As can be seen, the Initialization Protocol is faster than L-ZC in all the simulations and is 2-3 times as fast when the number of users is large. (This comparison ignores the fact that L-ZC converges to a state in which there are no collisions, but not necessarily to full efficiency: unless the size of the contention window happens to be precisely the number of users, it leaves gaps.)

N	10	20	30	40
IP+OP	0.9972	0.9935	0.9922	0.9900
L-ZC	0.2000	0.3995	0.5992	0.7945
PwM	0.4421	0.4149	0.4409	0.4464

TABLE IV

GOODPUT COMPARISON.

B. Single-Channel Scenarios

Speed of convergence is not the only relevant metric, of course; other metrics, such as goodput, are also important. Define normalized goodput to be the ratio of the number of successful transmissions to the total number of slots. In Table IV we show the results of a simulation in which we compare the normalized goodput (after 10,000 slots) of our Initialization Protocol followed by our Operating Protocol (IP+OP) against L-ZC [31] and Protocol with Memory (PwM) proposed in [29] for a simple single-channel scenario with a fixed number of users; for illustration we use $N = 10, 20, 30, 40$. As is clear, the improvement in goodput is dramatic.[6]

C. Multiple-Channel Scenarios with Channel Hopping

We now study goodput and delay performance of the Operating Protocol with $K = 0$ and $K = 1$ idle slots (assuming only Secondary Entrants). We consider a network with 3 channels; users will be able to hop between them. For the purpose of these simulations, we take the user hopping process to be exogenous. (We do not model the user's switching decision problem because we regard it as orthogonal to this paper. In practice the user hopping process should be determined by some channel exploration algorithm, e.g. [20]. Our focus is on the implications of the choice of Operating Protocol.) In this simulation, the user hopping process is simple: every T slots, a randomly chosen user switches to another channel. Thus, the parameter T reflects the exogenous user hopping rate. Our simulations highlight the trade-offs inherent in the choice of Operating Protocol (whether or not to leave an idle slot):

[6]As we have already noted, L-ZC achieves zero collisions but only achieves full efficiency if the size of the contention window happens to coincide with the number of users. PwM never achieves perfect coordination and only achieves zero collisions at the cost of extreme unfairness across users. In the simulation we use a contention window size of 50 for L-ZC and a fairness parameter of 0.5 for PwM.

- If $K = 0$ users transmit with no gaps so (in periods where there are no entrants) goodput is optimal; if $K = 1$, users transmit with gaps, so goodput is reduced.
- If $K = 0$, each entrant causes a number of collisions equal to the number of existing users on that channel which reduces goodput; if $K = 1$ no collisions occur, so goodput is increased,
- To which channel the user switches depends on the information available, which depends on the choice of Operating Protocol. When $K = 0$ the switching user has no information so chooses one of the other two channels at random; when $K = 1$ the switching user can count the number of users on each channel by observing idle slots and choose the channel with fewer users.

Figure 1 illustrates the normalized goodput for different average numbers of users per channel under different channel hopping frequencies. The solid curves represent the Operating Protocol with no idle slot and the dotted curves represent the Operating Protocol with 1 idle slot. Several points are worth noting.

- Given the average number of users per channel, goodput of the protocol with no idle slot increases when T increases/switching is less frequent. This is to be expected: switching causes collisions, and more frequent switching causes more collisions. However goodput for the protocol with one idle slot is almost independent of T: hopping does not cause collisions so loss of goodput is due almost entirely to the existence of an idle slot (and is smaller when there are fewer users per channel).
- Given the user hopping frequency, the goodput of the protocol with no idle slot decreases with the number of users. Again, this is expected because more users on a channel means more collisions. However goodput for the protocol with 1 idle slot increases with the number of users because the loss of goodput is mainly due to idle slots and idle slots occur less often.
- When the average number of users per channel is small, the protocol with no idle slot yields better goodput than the protocol with 1 idle slot, because the goodput loss created by the idle slots is relatively large. When the average number of users per channel is large, which protocol yields higher goodput depends on the user hopping frequency. If user hopping is rare, collisions in the protocol with no idle slot are rare and goodput is higher than in the

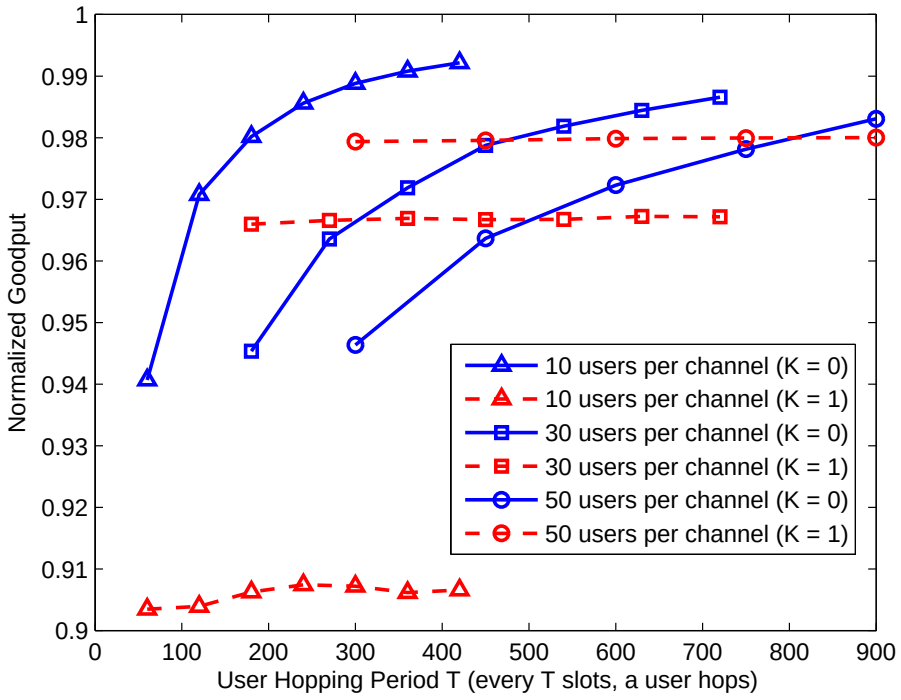

Fig. 1. Goodput performance with User Hopping.

protocol with 1 idle slot; when user hopping is frequent, collisions in the protocol with no idle slot are also frequent and goodput is lower than in the protocol with 1 idle slot.

In short: the designer's choice of protocol should depend on the parameters of the network.

Figure 2 and 3 illustrate the average delay and maximum delay, respectively. For these parameter values, there is little difference between protocols. However, when the number of users is small, the protocol with no idle slot achieves smaller average and maximal delay than the protocol with 1 idle slot, while when the number of users is large, the reverse is true. This reflects both the tradeoff between leaving slots idle and avoiding collisions and also the fact that, for the protocol with 1 idle slot, the switching user can choose the channel with fewer users, which reduces delay.

D. The Presence of Primary Users

In a cognitive system, it is important to protect the primary user from interruption by the secondary users. We measure interference experienced by the primary user by the collision probability of the primary users, i.e. the ratio of the number of slots in which the primary user experiences a collision to the number of slots in which it transmits. Of course the collision

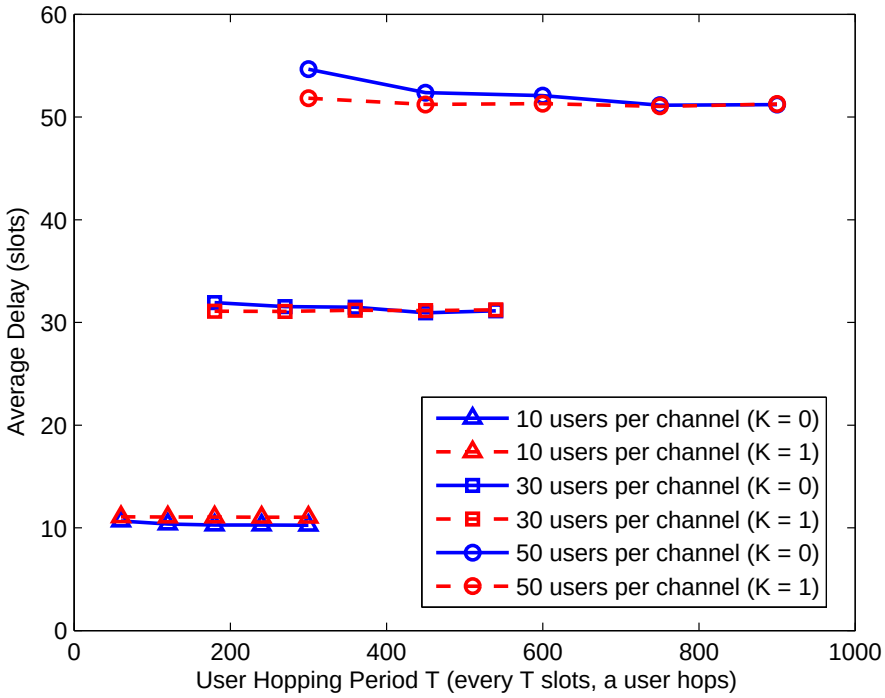

Fig. 2. Average delay performance with User Hopping.

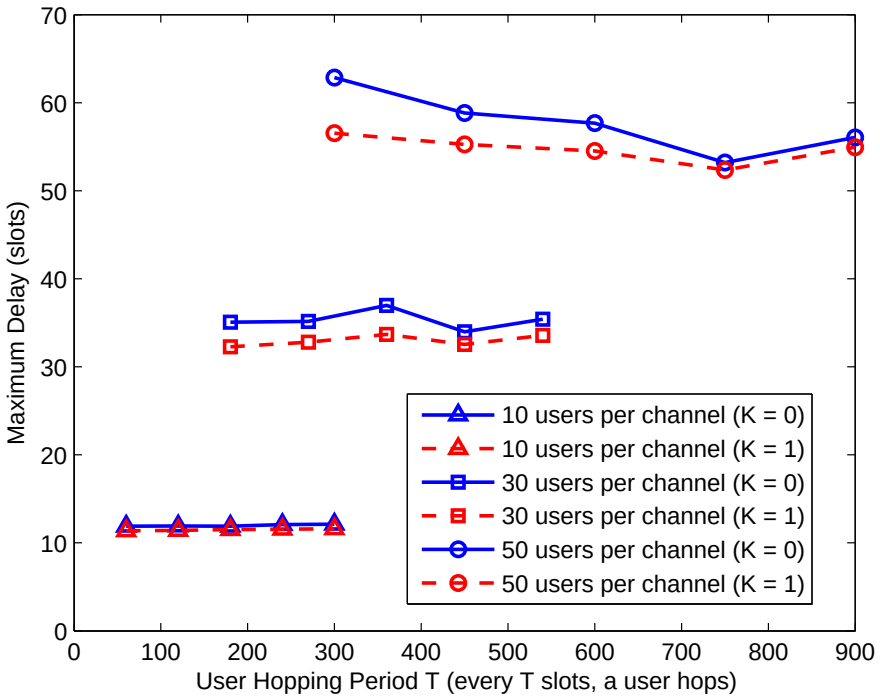

Fig. 3. Maximum delay performance with User Hopping.

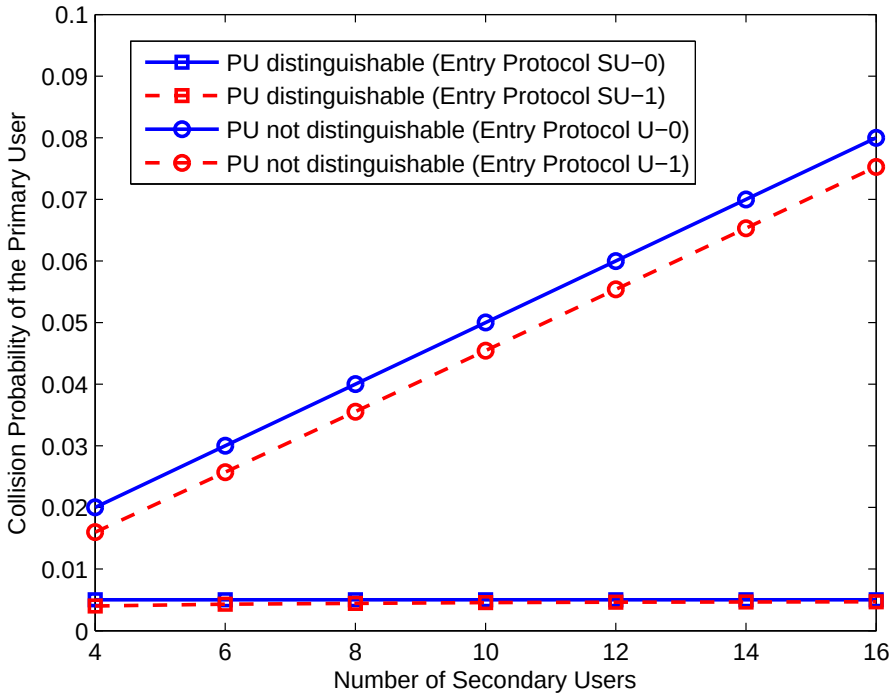

Fig. 4. Collision probability of the primary users.

probability of the primary user depends on its transmission pattern, e.g. the duration of its transmission. In realistic scenarios, this duration is much larger than the number of secondary users on the channel. In this simulation, we set the average number of transmission attempts by the primary user to be 200 slots in each duration.

Figure 4 shows the collision probabilities for various numbers of secondary users on the channel. When the primary user is physically distinguishable, Entry Protocol SU-0 and SU-1 are used. In SU-0, the only collision of the primary user occurs when it makes the first transmission attempt. After that, all secondary users detect the presence of the primary user and suspend their transmissions. In SU-1, there is a chance that the first slot that the primary user transmits in is an idle slot. The secondary users detect the presence of the primary user and suspend their transmissions and therefore, the primary user experience no collision. Hence, the collision probability of SU-1 is slightly lower than that of SU-0.

When the primary user is not physically distinguishable, Entry Protocol U-0 and U-1 are used. In U-0, all secondary users experience one collision with the primary user and then they detect the presence of the primary user. Therefore, the number of collisions experienced by the primary user is the number of secondary users on the channel. In U-1, there is a chance that the first slot

that the primary user transmits in is an idle slot. After one more slot, all users will know the presence of the primary user and quit transmitting. Therefore, the primary user experience no collision in this case. Hence, the collision probability of U-1 is slightly lower than that of U-0.

VI. Conclusions

In this paper, we have proposed a new class of MAC protocols for cognitive radios that deploy cooperative learning techniques enabling secondary users to achieve and maintain perfect coordination among themselves when accessing spectrum opportunities. Our proposed MAC protocols are completely distributed, requiring neither any central control nor any exchange of control messages between secondary users, fast and scalable. These protocols achieve optimal or near-optimal goodput with minimal or near-minimal delay, even when secondary users are frequently entering, exiting and switching between channels within a network . These protocols can be easily and efficiently configured to operate in scenarios where the primary users cannot be detected at the physical layer. In all these dimensions they outperform previous MAC protocols for cognitive radio networks.

References

[1] J. Mitola III, G. Maguire, "Cognitive radio: making software radios more personal," *Personal Communications*, IEEE 6(4) (1999) 13-18.

[2] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, F. Gringoli, "Wireless MAC processors: programming MAC protocols on commodity hardware," *IEEE INFOCOM 2012: 1269-1277*.

[3] W. Zame, J. Xu and M. van der Schaar, "Winning the Lottery: Learning Perfect Coordination with Minimal Feedback," *UCLA Technical Report*, 2012.

[4] K. Nakano and S. Olariu, "Randomized initialization protocols for ad hoc networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 11, no. 7, Jul. 2000.

[5] N. Abramson, "The Aloha system: another alternative for computer communications," in *Proc. the AFIPS Fall Joint Computer Communication*, vol. 37, pp. 281-285, 1970.

[6] L. G. Roberts, "Aloha packet system with and without slots and capture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 5, no. 2, pp. 28-42, Apr. 1975.

[7] W. Crowther, R. Rettberg, D. Walden, S. Ornstein, and F. Heart, "A system for broadcast communication: reservation-ALOHA," in *Proc. 6th Hawaii Int. Conf. Syst. Sci.*, Jan. 1973, pp. 371-374.

[8] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 3, pp. 535-547, Mar. 2000.

[9] IEEE 802.11a, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Highspeed Physical Layer in the 5 GHz Bang*, Supplement to IEEE 802.11 Standard, Sep. 1999.

[10] Federal Communications Commission, "Notice of proposed rule making: unlicensed operation in the TV broadcast bands," ET Docket No. 04-186 (FCC 04-113), May 2004.

[11] M. Marcus, "Unlicensed cognitive sharing of TV spectrum: the controversy at the federal communications commission," *IEEE Commun. Mag.*, vol. 43, no. 5, pp. 24-25, 2005.

[12] Y. Zhao, L. Morales, J. Gaeddert, K. K. Bae, J. -S. Um, and J. H. Reed, "Applying radio environment maps to cognitive wireless regional area networks," in *Proc. IEEE Int. Symposium on New Frontiers in Dynamic Spectrum Access Networks*, Apr. 2007.

[13] S. Shankar, C. Cordeiro, and K. Challapali, "Spectrum agile radios: utilization and sensing architectures," in *Proc. IEEE Int. Symposium on New Frontiers in Dynamic Spectrum Access Networks*, Nov. 2005.

[14] Y. Yuan, P. Bahl, R. Chandra, P. A. Chou, J. I. Ferrell, T. Moscibroda, S. Narlanka, and Y. Wu, "KNOWS: Cognitive radio networks over white spaces," in *Proc. IEEE Int. Symposium on New Frontiers in Dynamic Spectrum Access Networks*, Apr. 2007.

[15] H. Tang, "Some physical layer issues of wide-band cognitive radio systems," in *Proc. IEEE Int. Symposium on New Frontiers in Dynamic Spectrum Access Networks*, Nov. 2005.

[16] S. Mishar, R. Mahadevappa, and R. W. Brodersen, "Cognitive technology for ultra-wideband/WiMax coesistence," in *Proc. IEEE Int. Symposium on New Frontiers in Dynamic Spectrum Access Networks*, Nov. 2007.

[17] N. Khambekar, L. Dong, and V. Chaudhary, "Utilizing OFDM guard interval for spectrum sensing," in *Proc. IEEE Wireless Commun. And Networking Conf.*, Hong Kong, Mar. 2007.

[18] T. Yucek and H. Arslan, "Spectrum characterization for opportunistic cognitive radio systems," in *Proc. IEEE Military Commun. Conf.*, Oct. 2006.

[19] J. Jia, Q. Zhang, and X. Shen, "HC-MAC: A hardware-constrained cognitive MAC for efficient spectrum management," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 1, Jan. 2008.

[20] Q. Zhao, L. Tong, A. Swami and Y. Chen, "Decentralized cognitive MAC for opportunisitic spectrum access in ad hoc networks: a POMDP framework," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 3, Apr. 2007.

[21] A. Hsu, D. Wei, C. Kuo, "A cognitive mac protocol using statistical channel allocation for wireless ad-hoc networks," in *Proc. Wireless Communications and Networking Conference (WCNC)*, 2007.

[22] S. Huang, X. Liu and Z. Ding, "Opportunistic spectrum access in cognitive radio networks," *IEEE INFOCOM 2008*.

[23] J. Huang, R. Berry, and M. Honig, "Auction-based spectrum sharing," *ACM Journal on Mobile Networks and Applications*, vol. 11, issue 3, Jun. 2006.

[24] F. Fu and M. van der Schaar, "Learning to compete for resources in wireless stochastic games," *IEEE Trans. Veh. Tech.*, vol. 58, no. 4, 2009.

[25] A. Fattahi, F. Fu, M. van der Schaar, F. Paganini, "Mechanism-based resource allocation for multimedia transmission over spectrum agile wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 3, Apr. 2007.

[26] H. Wang, H. Qin, and L. Zhu, "A survey on MAC protocols for opportunistic spectrum access in cognitive radio networks," in *Int. Conf. Comput. Sci. Software Eng.*, 2008.

[27] T. V. Krishna and A. Dasa, "A survey on MAC protocols in OSA networks," *Comput. Networks*, vol. 53, no. 9, Jun. 2009.

[28] C. Cormio and K. R. Chowdhury, "A survey on MAC protocols for cognitive radio networks," *Ad Hoc Networks*, vol. 7, no. 7, Sep. 2009.

[29] J. Park and M. van der Schaar, "Cognitive MAC protocols using memory for distributed spectrum sharing under limited spectrum sensing," *IEEE Trans. Commun.*, vol. 59, no. 9, Sep. 2011.

[30] C-T Chou, S. Shankar, H. Kim, and K. Shin, "What and how much to gain by spectrum agility," *IEEE J. Sel. Areas in Commun.*, vol. 25, no. 3, Apr. 2007.

[31] M. Fang, D. Malong, K. R. Duffy and D. J. Keith, "Decentralised learning MACs for collision-free access in WLANS," *Wireless Networks*, DOI: 10.1007/s11276-012-0452-1, May 2012.

[32] J. Lee and J. Walrand, "Design and analysis of an asynchronous zero collision MAC protocol", Arxiv preprint 0806.3542, 2008.

APPENDIX

Here we give a more formal algorithmic description of the Initialization Protocol and prove the Theorem.

In the algorithm (below) each user $z \in Z$ keeps track of 2 variables $H(z), W(z)$ that describe the state of the system (the number of HITs and the number of WINs), two corresponding variables $h(z), w(z)$ that describe the personal state of the user (the number of HITs that have happened 'to' the user and the order in which WIN happened to the user) and a flag $f(z)$ that indicates, in the current cycle, whether the group to which the user belonged in the *previous* cycle contained other users, in which case $f(z) = 0$, or consisted only of the user, in which case $f(z) = 1$. These variables provide a summary of the past history of the user, and, taken together with observations of the current cycle, determine actions of the user in the current cycle. The protocol is designed so that users will have correct information about these variables (when necessary); in particular, users will always agree about the variables H, W that describe the state of the system.

We now turn to the proof of the theorem.

Proof of Theorem By construction, every cycle that ends in WIN results in an index for another user, so the algorithm will have converged as soon as N WIN cycles have occurred. Every cycle that ends in HIT partitions the set of active users into two non-empty subsets, so at most $N-1$ cycles can end in HIT. (We will use N for simplicity.) The remaining cycles end in either IDLE or NOISE and are "wasted". Hence we can obtain the probabilistic estimate we desire by accounting for cycles that end in WIN, cycles that end in HIT and cycles that are wasted and then accounting for the number of slots required for each of these cycles. Because cycles that end in IDLE require 1 slot and cycles that end in NOISE require 3 slots, we will get a sharper estimate if we account separately for cycles that end in IDLE and cycles that end in NOISE.

ALGORITHM 1: Initialization Protocol

Input: $H(z) = 1$; $W(z) = h(z) = w(z) = f(z) = 0$.

In each cycle:

if $H(z) \neq W(z)$, stop; else do the following.

if $w(z) > 0$ (retired) or $h(z) > 0$ (inactive): remain silent throughout the cycle

slot 1:

if $h(z) = 0$ and $f(z) = 1$: transmit

if $f(z) = 0$: randomize $R = 0.5T + 0.5S$

if $h(z) \neq= 0$: remain silent

switch channel realization of slot 1 **do**

case 0: **IDLE** do nothing, move to next cycle

case 1: **if** z's transmitted successfully in slot 1 set f(z) = 1

otherwise keep $f(z) = 0$

continue to slot 2

slot 2:

if $z \in Z_{RT}$: remain silent

if $z \in Z_{RS}$: transmit

switch channel realization of slot 1 and 2 **do**

case 11: **HIT**

if z transmitted successfully in slot 2, set $f(z) = 1$

otherwise, keep $f(z) = 1$

Maintain $h(z) = 0$, set $H(z) = H(z) + 1$, begin next cycle

case 10: continue to slot 3

slot 3:

if $z \in Z_T$ and $f(z) = 1$: transmit

if $z \in Z \backslash Z_T$: remain silent

switch channel realization of slot 1,2 and 3 **do**

case 100: **NOISE** do nothing, move to next cycle

case 101: **WIN**

All $z \in Z$ set $W(z) = W(z) + 1$, $h(z) = h(z) - 1$, begin next cycle

If $z \in Z_T$ set $w(z) = W(z)$, begin next cycle

To do the accounting, note first that cycles that end in WIN never involve randomization, while cycles that end in IDLE begin with at least two active users, all of whom randomize and obtain the realization S and cycles that end in IDLE begin with at least two active users, all of whom randomize and obtain the realization T . Because randomization is always $.5S + .5T$, the probability that all users obtain the same realization is at least $.5$, with equal probabilities that those randomizations are S or T. Our strategy is to estimate a number C of cycles C large enough so that the probability of at least N cycles ending in HIT is at least $1 - e^{-D}$ – this will be something on the order of $2N$ – *and* the probability of at least N of the $C - N$ cycles that do not end in HIT do end in IDLE is also at least $1 - e^{-D}$. After we have done this all that will remain is to count the corresponding number of slots.

The desired probability estimates will come out of the familiar Chernoff bound: if the probability of a success in one trial is (at least) $.5$ then the probability that the number of successes in K trials is less than $.5K - a$ is at most $e^{-2a/K}$. We want the probabilities of at least N HIT cycles (successes) in C HIT+IDLE+NOISE cycles (trials) total *and* the probability of at least $N/2$ IDLE cycles (short failures) in $C - N$ IDLE+NOISE cycles (total failures) to be at most e^{-D}. Applying the Chernoff bound and doing a little algebra we find that in the first instance we require

$$2[(C/2) - N]^2/C > D$$

which reduces to

$$C > 2N + D + 2(ND + D^2/4)^{1/2}$$

Applying the Chernoff bound in the second instance we find that we require

$$2[(C - N)/2 - (N/2)]^2/[C - N] > D$$

which reduces to:

$$C > 2N + D + 4(ND + D^2/4 - 8ND)^{1/2}$$

Taking the maximum, we see that it suffices if

$$C > 2N + D + 4(ND + D^2/4)^{1/2}$$

Now we count the number of slots. We need N cycles that end in WIN, which accounts for $3N$ slots and N cycles that end in HIT, which accounts for $2N$ slots. Of the cycles that do not

end in WIN or HIT, $N/2$ cycles end in IDLE, which accounts for $N/2$ slots and the remaining $C - N - N/2$ cycles end in NOISE, which accounts for $3(C - 3N/2)$ slots. Putting all this together, we it follows that if

$$K > 3N + 3N + (N/2) + 3[(N/2) + D + 4(ND + D^2/4)^{1/2}] = 7N + 3D + 12(ND + D^2/4)^{1/2}$$

then the probability of convergence in K slots is at least $1 - e^{-D}$ as asserted.

Some comments may helpful.

- The variable $h(z)$ keeps track of the transmission priority of the user: if $h(z) = 0$ the user will be active (transmit or randomize) in the current cycle; if $h(z) \neq 0$ the user will be inactive (silent).
- When a HIT occurs, all users observe it and adjust h according to their current status: users in Z_{RT} keep $h(z) = 0$, others lower $h(z)$ by 1.
- For an active user z, the flag $f(z)$ will be equal to 1 at the beginning of a cycle if and only if z transmitted successfully in the *previous* cycle. If $f(z) = 1$ at the beginning of a cycle then z will not randomize in that cycle but rather will follow the pattern *transmit, silent, transmit* .
- If $f(z) = 1$ at the beginning of a cycle for some active user z then there will be no other active users in that cycle and so necessarily the cycle outcome will be 101 WIN.
- If $f(z) = 0$ at the beginning of a cycle for all active users $z \in Z$ then the cycle outcome *cannot* be 101 WIN but must be 11 HIT or 0 EMPTY or 100 NOISE.
- The possible endings of each cycle – IDLE, HIT, WIN, NOISE – have distinct signatures of busy and idle, and hence can be observed and distinguished by all users (because all users can distinguish an idle channel from a busy channel) and hence all users can adjust the appropriate variables appropriately.