Cooperative Multi-Agent Learning and Coordination for Cognitive Radio Networks

William Zame, Jie Xu, and Mihaela van der Schaar

***Abstract*—The radio spectrum is a scarce resource. Cognitive radio stretches this resource by enabling secondary stations to operate in portions of the spectrum that are reserved for primary stations but not currently used by the primary stations. As it is whenever stations 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 stations with primary stations (so that secondary stations should not interfere with primary stations) and of secondary stations with each other. Coordination in this setting is especially challenging because of the various types of sensing errors. This paper proposes novel protocols that enable secondary stations to learn and teach with the goal of coordinating to achieve a round-robin Time Division Multiple Access (TDMA) schedule. 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 stations to learn from and condition on their own histories while simultaneously teaching other stations about these histories. Analytic results and simulations illustrate the power of these protocols.**

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

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 stations who need them for only a fraction of the time. Cognitive radio was introduced to address this problem, allowing secondary stations to access portions of the spectrum that are dedicated to primary stations 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 stations to detect the presence of primary stations 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 stations to coordinate with *each other* to share a given channel in a TDMA (Time Division Multiple Access) fashion. Without such coordination, there may be collision, congestion or interference, with concomitant loss of performance. Coordination problems are important even in the absence of primary stations (e.g., in cognitive radio networks operating in unlicensed bands such as 2.4GHz and 5GHz), and become even more important in the presence of primary stations, because secondary stations may frequently enter and exit channels (frequency bands) within the network or even the network itself. Such coordination can be obtained if the network is centrally controlled or if the stations can exchange control messages, but central control and the exchange of control messages are wasteful of resources and often impossible. This means that *distributed* protocols are extremely desirable.

In this paper, we propose protocols for efficient sharing of a frequency band. 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), *scalable* (adapting easily to both small and large numbers of stations) and *robust to errors*. The protocols we propose require only finite memory and can be formulated as finite automata, and do not require development of any new hardware. (See [1] for instance.)

We assume that the information stations obtain (including information about the actions of other stations) is limited and imperfect – we allow for observational *errors* – and our protocols are robust to such errors. A station that attempts to access the channel in a given period/slot will succeed and receive an acknowledgement if it is the only station attempting to access the channel in that period/slot – but acknowledgements may be lost (so that a station that is successful may think it was unsuccessful) and capture may occur (so that a station that is successful may think it was alone when it was not). A station that does not attempt to access the channel in a given period/slot will observe only whether the channel was idle (no other station attempted to access the channel) or busy (some other station attempted to access the channel) – but a station may observe a slot to be busy when it is idle or idle when it is busy (and in any case the station will not observe the *number* of other stations that attempted to access the channel in that slot). These informational assumptions seem to us to be the most realistic in the context of cognitive radio, but other

Manuscript received Nov. 17, 2012; revised Apr. 4, 2013 and Jun. 9, 2013.

W. Zame: Distinguished Professor, Departments of Economics and Mathematics, UCLA, Los Angeles, CA, 90095. Email: zame@econ.ucla.edu. Phone: (310) 206-9463. Fax: (310) 825-9528.

J. Xu: Corresponding Author. Department of Electrical Engineering, UCLA, Los Angeles, CA, 90095. Email: jiexu@ucla.edu. Phone: (310) 825-5843. Fax: (310) 206-4685.

M. van der Schaar: Professor, Department of Electrical Engineering, UCLA, Los Angeles, CA, 90095; Fellow, IEEE. Email: mihaela@ee.ucla.edu. Phone: (310) 825-5843. Fax: (310) 206-4685.

We are grateful to Giuseppe Bianchi for helpful discussions regarding errors in MAC protocols. Financial support was provided by National Science Foundation grants 0617027 and 1218136, 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.

assumptions have also been made in the literature ([2], [13], [14]) and might be sensible in other contexts. For instance, [2], [13] assume that *all stations* (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) – and that observations are *perfect*: there are no errors.

Both single-agent and multi-agent learning play a role in the existing literature on cognitive radio networks. In [3], the focus is on single-agent learning (the techniques come from Markov decision processes and from multi-armed bandits); in [17], [18] and [19], the focus is on multi-agent learning. In all of this work, the central problem addressed is to determine which of several channels (which frequency band) to access. In the former literature, stations learn characteristics of the system; in the latter literature, stations learn both characteristics of the system and (equilibrium) behavior of other stations. In the former literature, the end product of the learning process is simply a choice of channel; in the latter literature the end product of the learning process is an imperfectly coordinated profile of choices of channels – often a Nash or correlated equilibrium in which there are still collisions and empty slots. In the current paper, the focus is on multi-agent learning, but the central problem is on learning when – in which order – to access a *given* channel, and the end product of the learning process is *perfect coordination* with no collisions and no empty slots. In the literature, learning is distributed but individualistic; in the present paper, learning is distributed but cooperative: stations *learn* from their own actions but also use their actions to *teach* as well.

Some of the ideas exploited in this paper – randomization, collisions as a coordination device, and binary sort – are no doubt familiar from the literature. (E.g. the literature on slotted Aloha protocols and on exponential backoff protocols [4][5][6].) Others – silence as a message, cooperative learning, conditioning current actions on past history – are less familiar. 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 which stations use to teach other stations about themselves.

There is an extensive literature on MAC protocols for cognitive radio networks (for surveys see [11][12]), which can be divided into two distinct strands of research, addressing two distinct problems. The first strand of research focuses on how secondary stations can opportunistically identify the vacant portions of the spectrum and transmit in them while ensuring that the primary stations are minimally affected; representative work on this strand includes [7][8]. The second strand of research, to which this paper belongs. aims at developing protocols that enable secondary stations to coordinate with each other when accessing the spectrum opportunities that have been identified. A key challenge in this regard is how stations can/should optimally adapt their transmission strategies (including hopping among various frequency bands). Examples of work on this strand include [9] (which designs efficient auction mechanisms) and [10] (in which secondary stations 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 stations. These methods are costly and require much information exchange. Closer to our work is [13], which proposed a class of distributed MAC protocols with memory intended to achieve efficient spectrum sharing among secondary stations while protecting the primary station from potential interference by the secondary stations. As do we, [13] allows for the possibility that secondary stations cannot physically distinguish primary stations from secondary stations and that coordination messages cannot be exchanged between a station and a central controller or between stations, 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 stations past transmission histories, and this protocol does not guarantee that perfect coordination among secondary stations emerges either quickly or with high probability, and so fails to utilize the available spectrum capacity efficiently. 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.

The methods developed in this paper enable learning, teaching and coordination among anonymous users by signaling through actions – with no direct messaging. The current paper applies those methods to a specific problem in a specific setting, but these methods and appropriate extensions should be applicable to many other problems in many other settings; we discuss some of these in Section VII below. In particular, these methods may be useful when users are not fully anonymous, because they provide ways to exchange information by signaling with no direct messaging.

Following this Introduction, Section II formalizes the environment, the informational structure and the nature of our protocols. Sections III and IV present and analyze our proposed initialization protocols; Section V presents and analyzes our proposed re-coordination protocols. Section VI presents simulations and comparisons with existing protocols. Section VII concludes. Proofs and flowcharts are in the Appendix.

II. Framework

We consider the interaction among a finite set of stations Z over a (potentially) infinite time horizon. The true number of stations N is unknown, but bounded by $N^* < \infty$. Time is divided into *slots*, indexed by $t = 0, 1, \ldots$. In each slot, each station either transmits or remains silent; we use $1, 0$ (respectively) for these actions; $A = \{1, 0\}$ is the set of possible actions. If the actual number of stations is $N < N^*$, we treat the "missing" $N^* - N$ stations as if they were silent at all times, so A^{N^*} is the set of possible action profiles. Given an action profile, the *channel state* is the number of stations that transmit; the set of channel states is $\{0, 1, \ldots, N^*\}$.

A. Information

Stations do not directly observe the action profile or even the channel state but rather observe only a *signal* of the channel state; the signal observed depends (stochastically) on

the action of the station and on the true channel state. We assume here that a station that transmitted in a given slot observes (perhaps with error) only whether its transmission succeeded or failed and that a station that did not transmit in a given slot observes (perhaps with error) only whether the channel was busy or idle.[1] Hence the space of signals is $\Sigma = \{\text{SUCCESS}, \text{FAILURE}, \text{BUSY}, \text{IDLE}\}$. We write $\pi(\sigma|n,a)$ for the probability that a station observes the signal σ given that the true channel state is n and that the station took action a.[2] In view of our assumptions about what is observable, we have (for all channel states n)

$$\begin{array}{rcccl} \pi(\text{IDLE}|n,1) & = & \pi(\text{BUSY}|n,1) & = & 0 \\ \pi(\text{SUCCESS}|n,0) & = & \pi(\text{FAILURE}|n,0) & = & 0 \end{array}$$

We will discuss the kinds and probabilities of errors in Subsection II-D below.

B. Protocols

Stations condition current behavior on past observations. To formalize this, we need some definitions. A *personal history of length* $T \geq 0$ is an element of $\mathcal{H}_p(T) = A^T \times \Sigma^T$, specifying the actions taken and observations made at times $t = 0, \ldots, T$ for a single station. (The empty history is the unique history of length 0.) A *personal history* is an element of $\mathcal{H}_p = \bigcup_{T=0}^{\infty} \mathcal{H}_p(T)$. A *channel event* is an element of $E = (A \times \Sigma)^Z$: a profile of actions and observations by all stations. A *channel history of length* $T \geq 0$ is an element of $\mathcal{H}_c(T) = E^T$, specifying the profile of actions taken and observations made by all stations at times $t = 0, \ldots, T$. A *channel history* is an element of $\mathcal{H}_c = \bigcup_{T=0}^{\infty} \mathcal{H}_c(T)$. A channel history h is a sequence of profiles of actions and observations; for each station z let $O_z(h)$ be the sequence of observations of station z.

Write $\Delta(\{1,0\})$ for the set of probability distributions on $\{1,0\}$. We identify a probability distribution on $\{1,0\}$ with the probability weight it puts on 1, and hence identify $\Delta(\{1,0\})$ with $[0,1]$. Note that we identify 0 with remaining silent (for certain) and 1 with transmitting (for certain). A *protocol* is a function $F : \mathcal{H}_p \to \Delta(\{1,0\}) = [0,1]$ specifying, at each time, a (random) plan of action at that time, conditional on what the station observed at previous times: the date, the action of the station and the information obtained by the station. Keep in mind that the instruction is a *plan* of action (a probability $p \in [0,1]$); the *action taken* will be either transmit 1 or remain silent 0, depending on the realization of the randomization. Keep in mind as well that a protocol is a set of instructions for *an individual station*. Because stations are *ex ante* identical, we insist that the *same* protocol be specified for each station, but *stations randomize independently*, so *ex post*, different stations may experience different realizations, choose different actions and experience different histories. Note that a protocol F induces in the obvious way a unique probability distribution Π_F^T on $\mathcal{H}_c(T)$ for each T.

C. Convergence

The goal of our protocol is to coordinate the stations to produce TDMA/round-robin transmission; we make precise what it means for this to occur and the probability that it does occur.[3] Fix a protocol F. Consider a time T and a channel history $h \in \mathcal{H}_c(T)$. We say the protocol F has *converged (to TDMA/round-robin transmission) after* h if there is a one-to-one and onto mapping $i : Z \to \{1, \ldots, N\}$ such that if h' is a history of length $T + i(z) + kN$ that follows h (i.e., $h'(t) = h(t)$ for $t \leq T$) and that might occur when stations follow the protocol F (i.e., $\Pi_F^{T+i(z)+kN}(h') > 0$) it is the case that $F(O_z(h')) = 1$. That is, following the history h, each station transmits, in turn, once every N slots. Hence

$$\Pi_F^T\Big(\{h \in \mathcal{H}_c(T) : \ F \text{ has converged after } h \ \}\Big)$$

is the probability that convergence has occurred within T slots if stations follow the protocol F.

D. Errors

Various errors are conceivable in our environment; the following seem the most relevant:

- *Lost Ack* Some station transmits successfully but the acknowledgement is lost.
- *Misdetection/False Alarm* Some station perceives the channel as IDLE when it is actually BUSY or vice-versa.
- *Capture* At least two stations transmit but one signal is captured and acknowledged.

In the protocols we propose, Lost Acks are seen as collisions so simply cause the protocol to go through one more cycle of the Coordination phase; this slows but does not otherwise affect convergence. Misdetection/False Alarm and Capture are potentially more serious but analysis of our Basic Protocol (Section III) shows that it is robust to small probabilities of Misdetection/False Alarm or Capture and analysis of our Capture Protocol (Section IV) shows that it is robust to *arbitrary* probability of Capture, but convergence is slower.

III. The Basic Protocol

The proposed Basic Protocol is built around a binary sort. Of course, binary sorts and tree-splitting procedures have often been used in MAC protocols, but the cognitive radio environment limits the information available to stations, and this presents substantial difficulties. To see where these difficulties occur and what our protocol must accomplish to deal with them, suppose for the moment that the true number of stations N is known and that stations can observe the true channel state without error. In that setting the following protocol executes a binary sort. In the first step, all stations randomize equally between transmitting and remaining silent (choose $p = .5$). If all stations receive the same realization then all stations randomize again; otherwise, the realization partitions the set Z of stations into two disjoint non-empty subsets Z_0 (stations that remained silent) and Z_1 (stations that transmitted). (Because stations observe the true channel state, they all know which of these has occurred.) In the second step, stations in Z_0 remain silent (are inactive) and stations in Z_1 again randomize equally between remaining silent and transmitting, thereby partitioning Z_1 into Z_{10}, Z_{11}. Continuing in this way, we eventually reach a point where

[1] Equivalently: each station observes *whether* some other station transmitted but not the *number* of other stations that transmitted.

[2] We assume that signals are independent across periods; other assumptions are possible and could be accommodated.

[3] TDMA/round-robin transmission is optimal if only a single transmission can be successful in a single slot, which is the setting we study here.

$Z_{11\ldots11}$ is a singleton; assign this unique station the index $i = 1$. Now stations in $Z_{11\ldots10}$ (those that most recently randomized and remained silent) randomize equally between transmitting and remaining silent and proceed as before until reaching a singleton; assign this unique station the index $i = 2$. Continuing in this way leads eventually to a complete partition of Z into N singleton sets. Not counting steps in which all active stations receive the same realization (all transmit or all remain silent), this process requires exactly $N - 1$ steps.

However, even if there were no errors, in our environment this procedure would break down as soon as the first singleton set is reached. At that point, other stations would observe that the channel was busy but would not observe how many stations were transmitting; as a result, stations in $Z_{11\ldots10}$ would not know to resume activity and stations *not* in $Z_{11\ldots10}$ would not know to *not* resume activity. To overcome this difficulty, our protocol must provide information so that currently inactive stations can resume activity or remain inactive, as appropriate. Because we allow for the possibility that the true number of stations N is *not* known, our protocol must also provide information so that stations can determine that the sorting process has terminated. Because we allow for the possibility of errors, our protocol must provide information so that stations can detect and correct these errors. Finally, our protocol must provide this information in a completely distributed and anonymous way. Our protocol accomplishes all this by exploiting the ability of stations both to learn *and* to teach.

A. Individual Description of the Basic Protocol

The Basic Protocol is divided into phases: Initialization, Coordination, Synchronization, NS-Transmission, S-Transmission; some of these are in turn divided into modules and submodules. The protocol directs each station z to begin in the Initialization Phase and to set a collection of counters. After that, the protocol directs z to move to the Coordination Phase where it remains, cycling, until z believes perfect coordination is achieved or a prescribed number of slots has elapsed – whichever comes first. If z believes perfect coordination is achieved z moves to the NS-Transmission Phase; if the prescribed number of slots has elapsed, z moves to the Synchronization Phase. In the NS-Transmission Phase, the protocol directs station z to transmit in turn (in round-robin fashion) until a prescribed number of slots has elapsed and then to move to the Synchronization Phase. In the Synchronization Phase, z carries out a sync test. If z fails the test, it returns to the Initialization Phase and begins the entire protocol from the beginning; if z passes the test, it moves to the S-Transmission Phase. In the S-Transmission Phase, z transmits in turn (in round-robin fashion) for a certain number of cycles, then performs a *different* sync test. If z fails this latter sync test it returns to the Initialization Phase; if z passes this the test it remains in the S-Transmission Phase indefinitely. Note that the protocol enters the NS-Transmission Phase *before* passing through the Synchronization Phase and enters the S-Transmission Phase *after* passing through the Synchronization Phase – hence the terminology. From a system perspective the goals of the various Phases are: *Initialization* – set counters to keep track of the positions of stations and of the system; *Coordination* – count the number of stations, construct a complete ordering of the stations and inform each station of the number of stations and its place in this ordering; *Synchronization* – detect (some) errors in coordination; *NS-Transmission* – arrange transmissions in TDMA/round-robin fashion until the prescribed number of slots has elapsed; *S-Transmission* – detect (other) errors in coordination, arrange transmissions in TDMA/round-robin fashion, and transmit indefinitely. The Basic Protocol is presented in formal flow charts in the Appendix (Figure 6); we give below a description in words in terms of the instructions the protocol provides to an individual station z. The protocol makes use of a set of (positive integer) parameters that should be set by the designer/manager: D^* (the number of slots that elapse before the Synchronization Phase), r_1 (the number of repetitions required in the Synchronization Phase), r_2 (the number of repetitions (cycles) required in the S-Transmission Phase before applying the sync test, m_1 (the threshold at which conflicts in the Synchronization Phase trigger return to the Initialization Phase) and m_2 (the threshold at which conflicts in the S-Transmission Phase trigger return to the Initialization Phase). For a reason that will become clear later, we insist that $D^* > 5N$. Let $\ell = \lfloor 1 + \log_2 N^* \rfloor$ be the smallest integer such that $2^\ell > N^*$. The protocol begins in the Initialization Phase.

A few words about the sync tests may be helpful. We use the sync tests so that stations can determine whether there is agreement or conflict about the status of the system. Three kinds of disagreements/conflicts are possible. Most obviously, stations might disagree about whether coordination has been achieved (there are no stations remaining to be coordinated). Even if stations agree that coordination has been achieved, they might disagree about the number of stations (which is unknown at the beginning) and hence about the length of a round-robin cycle of the S-Transmission Phase. And even if they agree that coordination has been achieved and about the number of stations, their beliefs about their own positions in the ordering of stations might be in conflict. (For instance, two stations might each believe they are the first.) The first sync test (the Synchronization Phase) handles the first two kinds of disagreements/conflicts simultaneously; the second sync test (incorporated into the S-Transmission Phase) handles the third kind.[4] Our construction and arrangement of the two sync tests reflects the fact that, as the reader can easily see, a sequence of errors that leads to a disagreement of the first kind is much more likely than a sequence of errors that leads to a disagreement of the second kind, which is in turn much more likely than a sequence of errors that leads to a disagreement of the third kind. Incorporating the second sync test into the S-Transmission Phase is more efficient than conducting it as a separate phase because it does not waste slots when there is no disagreement of the third kind.

Initialization Phase Each station z maintains 5 global counters, 2 local counters and 1 indicator: $C(z)$ counts slots in the transmission phases; $D(z)$ counts the total number of slots; $H(z)$ counts the number of subsets into which the set of stations Z has been partitioned; $W(z)$ counts the number

[4]But note that, just as there might be errors during the Coordination Phase, there might also be errors during either or both sync tests. This is simply unavoidable.

of singleton sets in the current partition; $Y(z)$ counts duration in the S-Transmission phase; $h(z)$ labels the activity priority of station z (z is active when $h(z) = 0$ and further from becoming active when $h(z)$ is larger); $w(z)$ indicates whether station z belongs to a singleton set in the current partition (we say z is *completely partitioned*), and if so, in which order that happened (this will be the position in which station z transmits in the transmission phases); $y(z)$ counts the number of "errors" z has observed in the S-Transmission Phase; $f(z)$ indicates whether station z had been completely partitioned *in a previous slot* ($f(z)$ enables station z to teach *other* stations to adjust their priority counters). Initially: $H(z) = 1$ and $C(z) = D(z) = W(z) = h(z) = w(z) = f(z) = 0$. Note that $H(z) = W(z)$ exactly when station z believes the set of stations Z is completely partitioned into singletons but that $W(z) < H(z)$ before that point. If no errors have occurred all stations will have the same beliefs; if errors have occurred, stations may have differing beliefs; the Synchronization Phase detects possible differences in beliefs. After the Initialization is complete, the station moves to the Coordination Phase.

Coordination Phase The Coordination Phase is divided into modules and submodules. Within each module or submodule, each action requires 1 slot; after that action, the station *always* enters the Counting submodule, steps the counter $D(z)$ and then tests it: if $D(z) = D^*$ the station moves to the Synchronization Phase; otherwise it continues in the current module/submodule. This phase begins in the Start module.

- **Start module** The Start module determines whether the station z believes the system has become perfectly coordinated – if so, station z moves to the NS-Transmission Phase – and if not, whether z should be active or inactive, and if active, what action z should take.

– If $H(z) = W(z)$ then z believes all stations are perfectly coordinated and moves to the Transmission Phase; if $H(z) \neq W(z)$, station z advances in the Start module.

– If $w(z) > 0$ then z is completely partitioned and waits for the Coordination Phase to be complete; if $h(z) > 0$ then z waits for its activity priority to reach 0; in either case, z moves to the Waiting module. If $w(z) = h(z) = 0$ then z advances in the Start module.

– If $f(z) = 1$ then z has had a successful transmission in some previous cycle of the Coordination Phase, and moves to the Winning module; if $f(z) = 0$ then z moves to the Randomizing module.

- **Waiting module** Throughout this Module, station z is silent but senses the channel to learn the behavior of *other* stations, adjust counters appropriately and return to the Start module at the appropriate time.

– z is silent in the first slot and observes the channel state. If its observation is IDLE then (z believes that) *all* stations have been silent (either because they were inactive or because they were active and randomized but the realization of their randomization was to remain silent), and z returns to the Start Module to begin a new cycle. If the observation is BUSY then z is silent in the second slot and again observes the channel state. If this observation is BUSY then *some* stations have transmitted in the first slot and *other* stations have transmitted in the second slot, so the set of active stations has been partitioned; z steps its counters $H(z), h(z)$ and returns to the Start module to begin the next cycle. If this observation is IDLE, then z is silent in the third slot and again observes the channel state. If this observation is IDLE, then all stations that were active transmitted in the first slot and no station was successful; z simply returns to the Start module to begin the next cycle. If this observation is BUSY, then some station was successful in the first slot, so z steps $W(z)$ by 1, reduces $h(z)$ by 1 and returns to the Start module to begin the next cycle.

- **Winning module** In this module, $f(z) = 1$ which means station z transmitted successfully in some previous cycle and hence is completely partitioned; now z teaches *other* stations that z is completely partitioned, so that these other stations can adjust their counters appropriately. z accomplishes this by transmitting in the first slot, remaining silent in the second slot, transmitting in the third slot; after these actions, z adjusts its own counters appropriately and returns to the Start module to begin the next cycle.

- **Randomization module** This module and its submodules partition the set of active stations and maintain system-wide correct information about the partitioning. Station z begins by randomizing: transmit with probability .5, silent with probability .5. According to the realization, it moves to the Random-Silent or Random-Transmit submodule.

– **Random-Silent submodule** In this submodule, z has randomized and been silent in the first slot of this cycle; it then observes the channel state. If this observation is IDLE then (z believes that) *all* stations have been Silent (inactive stations are Silent as always; active stations randomized but received realizations of Silent) and z returns to the Start module to begin the next Step. If this observation is BUSY, then z transmits in the second slot and checks the result of the transmission. If the result is SUCCESS then z changes its indicator to $f(z) = 1$ (it is completely partitioned); whether it was successful or not, it adjusts $H(z), h(z)$ appropriately and returns to the Start module to begin the next cycle.

– **Random-Transmit submodule** In this submodule, station z has randomized and transmitted in the first slot of this cycle; it now checks the result of that transmission. If the result is SUCCESS, station z changes its indicator to $f(z) = 1$ (it is completely partitioned); whether successful or not, z is silent in the second slot and checks its observation of the channel state. If that observation is BUSY then a partitioning has occurred, the station steps $H(z)$ and returns to the Start module to begin the next cycle; if it was IDLE, no partitioning has occurred and z is silent in the third slot and returns to the Start module to begin the next cycle.

NS-Transmission Phase Station z remains silent until $w(z)-1$ slots have elapsed, then transmits. After that, z remains silent for $W(z)-1$ slots have elapsed and transmits again, and continues in this way until D^* slots (in total) have elapsed, then moves to the Synchronization Phase.

Synchronization Phase Station z takes actions that "send" – in binary – a sequence of numbers that reflect station z's opinion of the state of the system and observes whether what other stations send is in conflict.

- If $H(z) \neq W(z)$ station z believes the system *is not coordinated*. In this case, z is silent for $(r_1 + 1)\ell$ slots then

transmits for $(r_1+1)\ell$ slots; when these $2(r_1+1)\ell$ slots have elapsed, station z moves to the Initialization Phase and begins the protocol again.

- If $H(z) = W(z)$ station z (tentatively) believes the system *is coordinated* and has $W(z)$ total stations. In this case, station z "sends" $W(z)$ and repeats r_1 times, then "sends" $2^\ell - 1 - W(z)$ and repeats r_1 times; in each case station z "sends" all the leading zeroes so that each transmission is exactly ℓ slots long, representing ℓ binary digits. In all, this requires $2(r_1+1)\ell$ slots, the same number as in the previous case. In each slot, z sent one binary digit, first of $W(z)$ (r_1+1 times) and then of $2^\ell-1-W(z)$ (r_1+1 times). For each $k \leq \ell$, there are exactly $2(r_1+1)$ slots in which z sent the k-th digit. z observes the slots in which it sent 0 (was silent) – there are exactly r_1+1 such slots. Among those r_1+1 slots, z counts the number of slots in which it observed BUSY. If every other station z' agrees that $H(z') = W(z') = W(z)$, then every other station will be transmitting exactly when z is transmitting and will be silent exactly when z is silent, and hence for *every* k, z should never observe BUSY in *any* of these r_1+1 slots. If some other station z' does not agree that $H(z') = W(z') = W(z)$ then for *some* k, z should observe BUSY in *all* of these r_1+1 slots. If, for some k, z observes BUSY more than m_1 times (for the given threshold m_1) then z fails the sync test and returns to the Initialization Module; otherwise, z passes the sync test and moves to the S-Transmission Phase. (We use a threshold $m_1 > 1$ so that observational errors are less likely to cause the sync test to give the wrong answer.)

S-Transmission Phase Station z remains silent until $w(z)-1$ slots have elapsed, then transmits. After that, station z remains silent for $W(z)-1$ slots have elapsed and transmits again. Station z continues in this way for r_2 cycles. At that point it counts the maximum number of times that it has observed IDLE in *the same* slot. (If coordination is perfect, one station should be transmitting in each slot so z should observe no IDLE slots, but z might observe IDLE when it should observe BUSY.) If this number $y(Z)$ exceeds the threshold m_2 then z returns to the Initialization Phase; otherwise it remains in the S-Transmission Phase indefinitely. (We use a threshold m_2 for the same reason as above.)

B. System Description of the Basic Protocol

The previous Subsection describes the operation of the protocol from the point of view of an *individual station*; it may also be helpful to describe the operation of the protocol from the point of view of an *outside observer*. We begin by describing the Coordination Phase; for simplicity, we ignore errors in observing the channel state (but we discuss the impact of errors below). We first introduce some notation and terminology. Write Z_T, Z_S, Z_R for the sets of stations that are currently active and whose *instruction* is to transmit, remain silent, randomize (respectively) in the first slot of the current cycle, and Z_{RT}, Z_{RS} for the subsets of Z_R that are randomizing in this slot and whose *realizations* are transmit and remain silent respectively. Each station knows to which of these sets it belongs, but perhaps little or nothing else, even whether other sets are empty or not; much of our protocol is designed precisely to make these facts widely known. Table I distinguishes and gives names and codes to all possible combinations of cardinalities of the sets Z_T, Z_{RT}, Z_{RS} that might occur *when stations follow the protocol*.[5]

TABLE I
CODES, NAMES AND CARDINALITIES OF THE STATION SETS

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

The names and codes we have adopted are intended to be suggestive. A binary sort successively partitions stations into two groups until the remaining sets are all singletons. In our protocol, the partitioning is carried out endogenously by having stations randomize. HIT means that the current set of stations is partitioned in the current cycle (because some stations who were randomizing transmitted and others remained silent); EMPTY and NOISE mean that the current set of stations is not partitioned in the current cycle (because all stations who were randomizing experienced the same realization – silent in the first case, transmit in the second case). Finally, WIN means that a singleton has already been reached. (As we will see, this can only occur when the single station is *not* randomizing.) From the point of view of sorting, HIT represents a success, WIN represents the confirmation of a previous success (necessary in order to inform other stations), EMPTY and NOISE represent failures. (To estimate the probability of reaching the transmission phases we will eventually need to count successes and failures.) The codes we have adopted are also intended to be descriptive. If the first slot is EMPTY, the first slot will be idle, all stations will observe this and move to the next cycle. If the first slot is HIT, the first and second slots will be busy, all stations will observe that and move to the next cycle. If the first slot is NOISE, all stations will observe that transmission occurs in the first slot but that no transmissions occur in the second and third slots, and move to the next cycle. If the first slot is WIN, all stations will observe that transmission occurs in the first and third slot and that no transmission occurs in the second slot, and will then move to the next cycle.

We emphasize that the description above is from the point of view of an outside observer. After the first slot, a station can observe whether or not the channel is IDLE but nothing more. However, if in the first slot, the station observes that the channel is IDLE the protocol instructs it to move to the next cycle and if it observes that the channel is not IDLE the protocol instructs it to wait and learn what occurs in the second slot of the current cycle. If, in the second slot, the station observes that the channel is not IDLE the protocol instructs it to move to the next cycle and if it observes the channel is IDLE the protocol instructs it to wait and learn what occurs in the third slot in the current cycle.

By construction, the protocol instructs the stations to cycle repeatedly through the Coordination Phase until $H(z) = W(z)$ and then to move to the NS-Transmission Phase. If no errors occur, all stations agree on the values of H, W so all stations move to the NS-Transmission Phase at the same time. But if errors do occur, different stations may make

[5]Other combinations might occur if stations do not follow the protocol but are irrelevant.

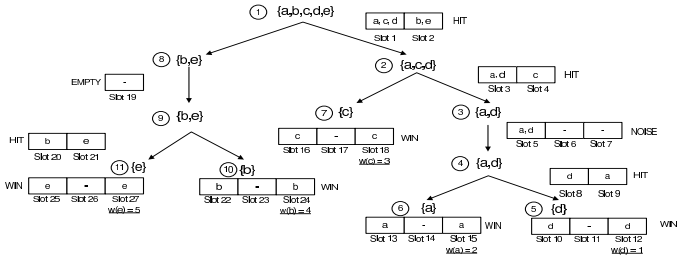

Fig. 1. Schematic: Operation of the Coordination Phase with Stations $\{a, b, c, d, e\}$

contradictory observations about the channel state, and hence may not agree on the values of H, W so may not move to the NS-Transmission Phase at the same point – or at all. However, after D^* slots have elapsed all stations move to the Synchronization Phase, during which they check their beliefs about H, W. If they agree that $H = W$ and on the common value, they move to the S-Transmission phase; if not they return to the Initialization Phase and begin the protocol anew.

We incorporate two different transmission phases because it is more efficient. If stations moved directly to the S-Transmission Phase whenever they believed $H(z) = W(z)$ and remained there indefinitely, without a sync test, there would be some non-vanishing probability that the protocol would never converge; if stations did not move to the NS-Transmission Phase as soon as they believed $H(z) = W(z)$ they would be "wasting" many opportunities for perfectly coordinated transmission.

Figure 1 shows in schematic form an example of the Coordination Phase with 5 stations $\{a, b, c, d, e\}$. Each node represents a cycle. Node 1 is a HIT cycle: in slot 1 all stations randomize; $\{a, c, d\}$ transmit, $\{b, e\}$ are silent; in slot 2 $\{a, c, d\}$ are silent, $\{b, e\}$ transmit. Node 2 is another HIT cycle (comprising slots 3,4). Node 3 is a NOISE cycle (comprising slots 5,6,7), etc.

C. Convergence Probability and Speed

Our object here is to provide an analytic estimate for the probability that the proposed protocol converges to TDMA/Round-Robin Transmission in a given number of slots. We first estimate the probabilities that the protocol goes awry in either of the two sync tests. In the Synchronization Phase, this could happen in one of two ways: either (i) the system *is* coordinated (i.e., $H(z) = W(z)$ for all stations z and $W(z) = W(z')$ for all stations z, z') but at least one station believes the system *is not* coordinated (i.e., fails the synchronization check); or (ii) the system *is not* coordinated (i.e., either $H(z) \neq W(z)$ for some stations z or $W(z) \neq W(z')$ for some stations z, z') but at least one station believes the system *is* coordinated (i.e., passes the synchronization check). In the S-Transmission Phase, this could again happen in one of two ways: either (iii) the ordering of stations is consistent (for all stations $z \neq z'$ it is the case that $w(z) \neq w(z')$) but some station believes it to be inconsistent, or (iv) the ordering of stations is inconsistent (for some stations $z \neq z'$ it is the case that $w(z) = w(z')$) but all stations believe it to be consistent. If bounds on the probabilities of mis-detection/false alarm are known, we can estimate the probabilities of each of (i), (ii), (iii), (iv) and show that is possible to choose the parameters $r_1, m_1; r_2, m_2$ so that each has probability less than any pre-specified positive quantity. The formulation of these estimates is in terms of the binomial distribution: given k, K, γ write $B(k, K; \gamma)$ for the probability of obtaining at least k successes in K independent Bernoulli trials when the probability of success in each trial is γ; the proofs of the following Lemmas and the Theorem are in the Appendix.

Lemma 1 Fix r_1, m_1. Assume $\pi(\text{BUSY}|0, 0) < \varepsilon$ (i.e., the probability of false alarm is less than ε). If the system *is* coordinated then the probability that at least one station fails the sync test (i.e., believes the system *is not* coordinated) is at most $N\ell B(m_1, (r_1 + 1); \varepsilon)$.

Lemma 2 Fix r_1, m_1. Assume $\pi(\text{IDLE}|n, 0) < \varepsilon$ for each $n > 0$ (i.e., the probability of mis-detection is less then ε). If the system is *not* coordinated then the probability that at least one station passes the sync test (i.e., believes the system *is* coordinated) is at most $N[1 - B(m_1, (r_1 + 1); 1 - \varepsilon)]$.

Lemma 3 Fix r_2, m_2. Assume $\pi(\text{BUSY}|0, 0) < \varepsilon$ (i.e., the probability of false alarm is less than ε). If the ordering *is* consistent then the probability that at least one station believes the ordering *is not* consistent is at most $NB(m_2, r_2; \varepsilon)$.

Lemma 4 Fix r_2, m_2. Assume $\pi(\text{IDLE}|n, 0) < \varepsilon$ for each $n > 0$ (i.e., the probability of mis-detection is less then ε). If the ordering is *not* consistent then the probability that at least one station believes the ordering *is* consistent is at most $N[1 - B(m_2, r_2; 1 - \varepsilon)]$.

To minimize the possibility that the sync tests produce the wrong result for some station we should choose $r_1, m_1; r_2, m_2$ to make the upper bounds in Lemmas 1-4 small. Since the upper bound N^* for the number of stations and $\ell = \lfloor 1 + \log_2 N^* \rfloor$ are fixed, we can make all of them small at the same time by choosing r_1, r_2 large and $m_i = \lfloor 1 + r_i/2 \rfloor$ but this is not the most efficient choice. Hence we set:

$$\begin{aligned} \delta \;=\; 1 - \max_{m_1, m_2} \Big\{ & N^* \lfloor 1 + \log_2 N^* \rfloor B(m_1, \ell(r_1 + 1); \varepsilon) \;, \\ & N^*[1 - B(m_1, (r_1 + 1); 1 - \varepsilon)], \\ & N^* B(m_2, r_2; \varepsilon), \\ & N^*[1 - B(m_2, r_2; 1 - \varepsilon)] \Big\} \end{aligned}$$

Note that δ depends on N^*, ε (which are characteristics of the environment) and on r_1, r_2 (which are parameters chosen by the designer/manager. In view of Lemmas 1-4, δ is a lower bound for the probability that the sync tests produces the right results when we use the optimal thresholds m_1, m_2. Assuming $\varepsilon < .5$, it is easy to check that $\lim_{r \to \infty} \delta = 1$ for each fixed N^*, ε: repetition guarantees that the probability the sync test gives the right result is as close to 1 as we like.

We now turn to our analytic estimate for the speed and probability of convergence (Theorem 1). The statement is a bit complicated so some prior explanation may be useful. First, note that by definition and construction of the protocol, convergence means that the protocol has gone through the Synchronization Phase, that all stations have passed the sync test (i.e., all stations agree that $H(z) = W(z)$ and agree on the common value of these counters), and that all stations are correct. However, the protocol does not enter the Synchronization phase until D^* slots have elapsed and the Synchronization phase itself requires $2\ell(r_1 + 1)$ slots, so convergence *cannot* occur before $D^* + 2\ell(r_1 + 1)$ slots have elapsed. If the protocol

has gone through the Synchronization phase and all stations have *failed* the sync test then the protocol returns to the Initialization Phase and begins again, so the protocol cannot possibly converge until at least *another* $D^* + 2\ell(r_1 + 1)$ slots have elapsed, and so forth. Moreover, if the protocol has passed the Synchronization Phase but failed the sync test in the the S-Transmission phase this requires another $r_2 N$ slots. Second, note that any errors in the Coordination Phase might lead to failure of the system to be synchronized; the easiest way to take this into account is simply to give a lower bound on the the probability that *no errors occur* during the Coordination Phase (in a particular round). However, we also have to worry about errors of synchronization, and in some sense those errors are more problematic since if stations disagree about the results of the sync tests they will *never* recover. However, as we have already noted, no matter what ε is, the probability that this occurs can be made arbitrarily small by appropriate choice of the parameters D^*, r_1, r_2 (but it *cannot be* made equal to 0).

We will find a lower bound α for the probability that after D^* slots of the Coordination Phase, all stations agree that $H(z) = W(z)$ and agree on the common value (call this *coordination*) conditional on no errors having occurred and a lower bound β for the probability that no errors occur during the Coordination Phase in that round; we have already found a lower bound δ for the probability that the stations all receive the correct result from the sync test. It follows that the probability of convergence in one round is at least $\alpha\beta\delta$, and it can be shown that the probability of convergence in one or two rounds is at least $\alpha\beta\delta + (1-\alpha\beta)\alpha\beta\delta^2$, and so on; this is our analytic estimate.

Theorem 1 Fix N^* (the maximum possible number of stations) and the parameters D^*, r. If

$$\begin{array}{rclcrcl} \pi(\text{BUSY}|0,0) & < & \varepsilon & , & \pi(\text{FAILURE}|1,1) & < & \varepsilon \\ \pi(\text{IDLE}|m,0) & < & \varepsilon & , & \pi(\text{SUCCESS}|n,1) & < & \varepsilon \end{array}$$

for each $m > 0, n > 1$ (i.e., all the error probabilities are less than ε) then, for the Basic Protocol:

1) the probability of coordination in one round (D^* slots) of the Coordination phase, conditional on no errors occurring, is at least $\alpha = 1 - 2\exp\left(-\frac{2(D^*-7N)^2}{3(D^*-4N)}\right)$;
2) the probability of no errors occurring during one round of the Coordination Phase is at least $\beta = 1 - D^* N\varepsilon$;
3) the probability of convergence in one round of the Basic Protocol ($D^* + 2\ell(r_1+1)$ slots) is at least $\alpha\beta\delta$;
4) for $R \geq 2$ the probability of convergence in at most R rounds of the Basic Protocol ($R[D^*+2\ell(r_1+1)]+(R-1)r_2 N$ slots) is at least $\alpha\beta\delta+(1-\alpha\beta)\alpha\beta\delta^2+\ldots+(1-\alpha\beta)^{R-1}\alpha\beta\delta^R$.

As Table II shows, Theorem 1 gives very fast rates of convergence – but the actual rates of convergence (from simulations) are even better. One reason is that Theorem 1 treats all errors as equal and catastrophic – but not all errors are equal (loss of acknowledgement creates only minor delay in convergence, not collapse of the protocol) and not all errors are catastrophic (some errors may be rectified by later errors). Moreover, the proof of Theorem 1 treats the probability that any given cycle of the Coordination Phase ends in HIT as .5 – but if the number of active users exceeds 2 this probability is at least .75.

Prob.	Method	N = 3	N = 5	N = 10	N = 20
0.9	Est.	60	76	115	189
	Sim.	52	60	98	156
0.99	Est.	65	81	120	197
	Sim.	58	75	108	177
0.999	Est.	69	85	125	203
	Sim.	65	82	117	185

TABLE II
ESTIMATES AND SIMULATIONS OF CONVERGENCE SPEED ($\varepsilon = 10^{-5}$)

IV. THE CAPTURE PROTOCOL

The Basic Protocol makes no special provision for capture (at least 2 stations transmit but one transmission is "captured" and acknowledged), as distinct from other types of errors. However, in some environments, capture has much higher probability than the other types of errors we have discussed. In such environments, the Basic Protocol might require *many* rounds to converge. It seems desirable therefore to propose a modification on the Basic Protocol that handles capture smoothly. The idea of the modification is simple: whenever a cycle of the Coordination Phase does not terminate after the first slot (i.e., when the first slot is not EMPTY), the stations will convey additional information to confirm/disconfirm whether a successful transmission resulted from a single station transmitting in a particular slot or a capture. This will require 4 slots – instead of 2 or 3. We can most easily describe the modification by classifying behavior according to the number of currently active stations.

- *There is exactly 1 active station.* The station that is active knows it is alone in its subset of the partition. This station takes the actions 1,0,1,1 (Transmit, Silent, Transmit, Transmit). All stations observe 1011, interpret this as WIN and adjust counters accordingly.
- *There are at least 2 active stations.* All active stations randomize in the first slot. The remainder of the step depends on what happens in the first slot

– *All stations are silent in the first slot.* The outcome of this step will be 0; all stations observe this, adjust the counter D (only) and then move directly to the Start module.

– *All stations transmit in the first slot.* All stations are silent in the second and fourth slots. Any station that observed SUCCESS in the first slot is silent in the third slot; every other active station observed COLLISION in the first slot and transmits in the third slot. (Because there are at least two active stations, the observation of SUCCESS must necessarily be the result of a capture.) The outcome of this step must be 1010; all stations observe this and interpret that there has been a further partition of the set of stations, and adjust the counters D, H, h accordingly.

– *At least one agent transmits in the first slot and at least one agent is silent in the first slot.* Stations that transmitted in the first slot and observed SUCCESS in the first slot are silent in the third slot, stations that transmitted in the first slot and observed COLLISION transmit in the third slot; all stations that transmitted in the first slot are silent in the fourth slot. Stations that transmitted in the second slot and observed SUCCESS in the second slot are silent in the fourth slot, stations that transmitted in the second slot and

observed COLLISION transmit in the fourth slot; all stations that transmitted in the second slot are silent in the third slot. A station that observed a SUCCESS in the first (second) slot and IDLE in the third (fourth) slot knows that the SUCCESS was real (no other station was transmitting) and interprets the SUCCESS as an error and adjusts its flag; a station that observed a SUCCESS in the first (second) slot and BUSY in the third (fourth) slot knows that the SUCCESS was a capture (some other station was transmitting) and does not adjust its flag. All stations that were not active observe 11xx and increase their H, h counters by 1.

- If *some* stations are randomizing and *all* stations transmit in the first slot, then *all* stations will be silent in the second slot. When this occurs, *all* stations know that a partition did *not* occur and that any acknowledgement received in the first slot must have been an accidental capture (because stations that transmitted in the first slot were randomizing, hence not alone in their subset of the partition). The capture is ignored and all stations will be silent again in the third slot. All stations will observe this step to be NOISE.
- If *some* stations are randomizing, *some* stations transmit in the first slot and *some* stations are silent in the first slot, then *some* stations will transmit in the second slot. When this occurs, *all* stations know that a partition *did* occur. Now stations must learn whether captures that occurred in the first or second slots were accidental captures; to accomplish this, stations that transmitted in the first slot and received an acknowledgement will be *silent* in the third slot, while stations that transmitted in the first slot and did *not* receive an acknowledgement will *transmit* in the third slot; stations that transmitted in the first slot and received an acknowledgement but observe BUSY in the third will learn they experienced a false capture and will ignore it. Similarly, stations that transmitted in the second slot and received an acknowledgement will be *silent* in the fourth slot, while stations that transmitted in the second slot and did *not* receive an acknowledgement will *transmit* in the fourth slot; stations that transmitted in the second slot and received an acknowledgement but observe BUSY in the fourth slot will learn they experienced a false capture and will ignore it.

The analytic estimate for convergence of the capture protocol (Theorem 2) is almost the same as for the Basic Protocol: all that changes is that the various cycles are longer. To avoid confusion, we use $\overline{D}$ for the number of slots that must elapse before entering the Synchronization Phase.

Theorem 2 Fix N^* (the maximum possible number of stations) and the parameters $\overline{D}, r_1, r_2$. If

$$\pi(\text{BUSY}|0,0) < \varepsilon \quad , \quad \pi(\text{FAILURE}|1,1) < \varepsilon$$
$$\pi(\text{IDLE}|m,0) < \varepsilon \quad , \quad \pi(\text{SUCCESS}|n,1) < \varepsilon$$

for each $m > 0, n > 1$ (i.e., all the error probabilities are less than ε) then, for the Capture Protocol:

1) the probability of coordination in one round of the Coordination Phase ($\overline{D}$ slots), conditional on no errors occurring, is at least $\alpha = 1 - 2\exp\left(-\frac{(\overline{D}-(21N/2))^2}{2(\overline{D}-(17N/2))}\right)$;

2) the probability of no errors occurring in one round of the Coordination Phase is at least $\beta = 1 - \overline{D}N\varepsilon$;

3) the probability of convergence in one round ($\overline{D}+2\ell(r+1)$ slots) of the Capture Protocol is at least $\alpha\beta\delta$;

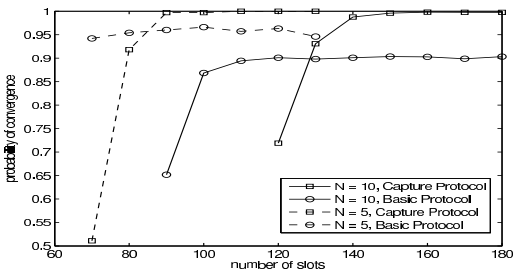

Fig. 2. Convergence of the Capture Protocol (Capture Probability = .01)

4) for $R \geq 2$ the probability of convergence in at most R rounds of the Capture Protocol ($R[\overline{D} + 2\ell(r_1 + 1)] + (R - 1)r_2N]$ slots) is at least $\alpha\beta\delta + (1 - \alpha\beta)\alpha\beta\delta^2 + \ldots + (1 - \alpha\beta)^{R-1}\alpha\beta\delta^R$.

The Capture Protocol is immune to captures (because captures are always detected and corrected) and hence the speed of convergence is independent of the capture probability. (However, because HIT, NOISE, and WIN cycles are longer in the Capture Protocol than in the Basic Protocol – 4 slots in the Capture Protocol; 2, 3 and 3 slots in the Basic Protocol – the Capture Protocol is approximately roughly 50% slower than the Basic Protocol and the probabilities of Misdetection/False Alarm are correspondingly higher.) Figure 2 illustrates performance of the Basic Protocol and the Capture Protocol in a setting where capture is relatively likely (probability = .01): as can be seen, the Capture Protocol converges much more quickly than the Basic Protocol.

V. Entry

The initialization protocol coordinates secondary stations. If secondary stations always entered the network at the same time, the protocols proposed above are all that would be required. However this is seldom the case: secondary stations arrive (and depart) at need. There are many different cognitive radio environments and different environments may call for different entry protocols: an entry protocol that works smoothly in a single-channel environment may not work smoothly in a multi-channel environment; an entry protocol that works smoothly in an environment in which when primary stations can be recognized by their signatures may not work at all in an environment in which primary stations can not be recognized by their signatures [8]. Here we offer only a single, simple entry protocol but many others could be constructed using the same principles. The entry protocol is typically *much* faster than the coordination protocol(s) proposed above so it is more efficient to employ this entry protocol when possible rather than abandoning the coordination that already exists and re-coordinating from scratch.

In what follows we assume that primary users can be recognized by their signatures (so only entry of secondary users needs to be considered), that entry does not occur too often (so at most one entry occurs during the coordination phase of the entry protocol), and that entry occurs at a point in which Current Stations are perfectly coordinated and transmitting in a round-robin/TDMA fashion with no gaps. In particular, we assume that, at the point just before the Entrant attempts to enter: (1) all Current Stations stations know the number N of stations; (2) all Current Stations know the order

in which they should transmit; (3) all Current Stations transmit in turn.

In order for an additional station to enter and the resulting $N+1$ stations to again become perfectly coordinated, three things must happen: Current Stations must learn that there is an Entrant, Current Stations must "make space" for the entrant, and the Entrant must learn to occupy that space. In order for a Current Station to learn that there is an entrant, something must happen that a current station does not expect: the only thing this could be is a collision. Because stations who do not transmit cannot distinguish between a successful transmission and a collision, *each* of the Current Stations must experience a collision. Having experienced a collision, the Current Stations can "make space" by waiting one additional slot. In order for an Entrant to learn to occupy the space created, something must happen that the Entrant does not expect: the only thing this could be is a success. But one success is not enough; the Entrant must experience two successes, so that it learns the number of Current Stations (as the length of the interval between these two successes). The Entry Protocol arranges these events in a convenient way. We describe the protocol in an environment in which there are no errors; it can be made robust to errors by appending a synchronization check, as in the Basic Protocol.

Entry Protocol

- *Entrant* The Entrant e keeps an indicator $g(e)$ and a counter $G(e)$. Initially $g(e) = G(e) = 0$. While $g(e) = 0$ the Entrant transmits in *every slot* until it experiences one successful transmission, at which point it sets $g(e) := 1$. While $g(e) = 1$ the Entrant continues to transmit until it experiences a collision, at which point it sets $g(e) := 2$ and $G(e) := 1$. While $g(e) = 2$ the Entrant continues to transmit and steps the counter $G(e)$ in each slot, until it experiences a second successful transmission, at which point it sets $g(e) := 3$. The Entrant then remains silent for $G(e)$ slots (note that $G(e)$ is the number of successive collisions experienced by the Entrant after the first successful transmission, which will be precisely the number of Current Stations), then transmits, then is silent for $G(e)$ slots, etc.
- *Current Station* Each Current Station transmits in turn every N slots until experiencing a collision, which reveals that a new station (necessarily a secondary station in this case) has appeared. Following a collision, the Current Station waits one extra slot before transmitting again; i.e., following a collision each Current Station transmits every $N+1$ slots.

If there are N Current Stations and the Entrant enters in slot k, there will be N successive collisions, followed by 1 successful transmission by the Entrant, followed by N more successive collisions – a total of $2N+1$ coordinating slots. After that, the Entrant and the Current Stations will be perfectly coordinated: in slot $k+2N+1$ the Entrant will transmit, followed by the N Current Stations, followed by the Entrant, etc. See Figure 3 for a simple schematic diagram.

VI. Simulations

A. Convergence Speed of the Basic Protocol

We have already reported in Table II the result of simulations that compares the actual speed of convergence of the Basic Protocol with the estimate provided by Theorem 1 for various values of $N = 3, 5, 10, 20$ and assuming $\varepsilon = 10^{-5}$. Figure 4 shows the results of simulations for $\varepsilon = 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}$.[6] As the reader will see, the Basic Protocol converges quickly – so long as the product of the error probability, the number of users and the checking interval D^* (roughly, the probability that *some* station makes an error during the Coordination phase) is not too big.

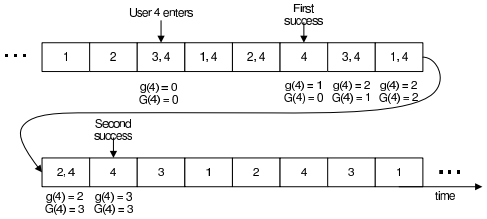

Fig. 3. Operation of the Entry Protocol

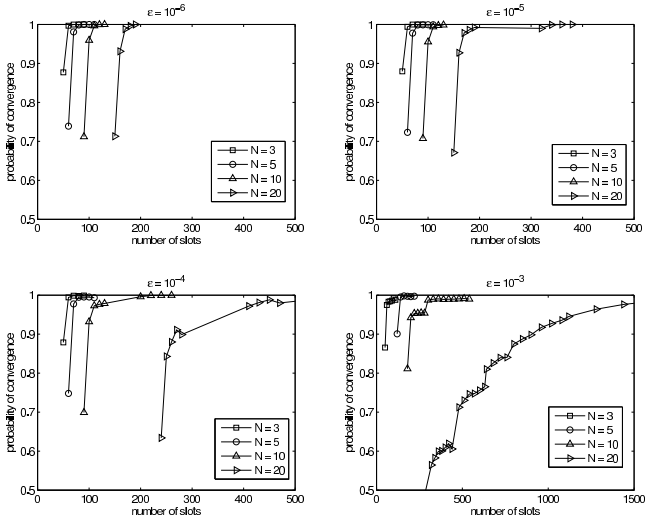

Fig. 4. Convergence Speeds.

Table III compares (by simulation) speed of convergence of our Basic Protocol with the L-ZC (Zero Collision with Learning) protocol proposed in [15] (as an improvement on the ZC (Zero Collision) protocol proposed in [16]). As will be seen, the Basic Protocol is comparably fast or faster – even though the Basic Protocol allows for errors and achieves round-robin/TDMA transmission, which L-ZC (and other existing protocols) do not.

B. Goodput

Speed of convergence is not the only relevant metric, of course; other metrics, such as goodput, are also important. We 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 [15] and Protocol with Memory (PwM) proposed in [13] for a simple single-channel scenario with a fixed number of stations. As is clear, the improvement in goodput is dramatic.[7]

[6] Suitable values for D^* can be computed from Theorem 1 or by simulation.

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

Probability		0.9	0.99	0.999
N=3	BP	52	58	65
	L-ZC	16	29	38
N=5	BP	60	75	82
	L-ZC	32	54	74
N=10	BP	98	108	117
	L-ZC	77	118	165
N=20	BP	156	177	185
	L-ZC	178	263	348

TABLE III
COMPARISON OF CONVERGENCE SPEEDS

	N = 3	N = 5	N = 10	N = 20
BP	1.00	1.00	1.00	1.00
L-ZC	0.15	0.25	0.50	1.00
PwM	0.44	0.41	0.43	0.43

TABLE IV
GOODPUT COMPARISON.

C. Channel Hopping

As a final simulation, we consider goodput performance of the Entry Protocol in a setting in which secondary stations hop among 3 channels in a network. For the purpose of these simulations, we take the station hopping process to be exogenous: every T slots, a randomly chosen station switches to another channel.[8] Figure 5 illustrates the normalized goodput for different average numbers of stations per channel under different channel hopping frequencies.

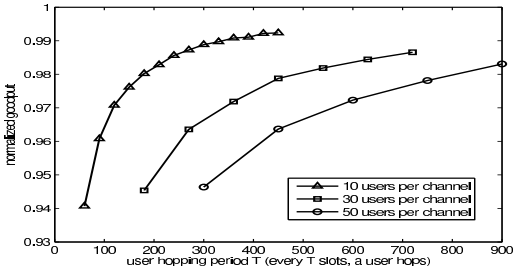

Fig. 5. Goodput Station Hopping.

VII. CONCLUSIONS AND FURTHER DIRECTIONS

In this paper, we have proposed a new class of MAC protocols based on cooperative learning that enables secondary stations to achieve and maintain perfect coordination. Our proposed MAC protocols are completely distributed, requiring neither any central control nor any exchange of control messages between secondary stations, fast, scalable and robust to many kinds of errors. In the present context, perfect coordination makes it possible for users to share a channel in a TDMA/round-robin fashion. More generally, perfect coordination can make it possible for stations to efficiently share specific resources – not necessarily slots.

As noted in the Introduction, the methods developed here enable learning, teaching and coordination among anonymous users by signaling through actions with no direct messaging. The current paper applies those methods to a specific problem in a specific setting. However, because these methods provide ways to exchange information by signaling through actions with no direct messaging, they may be applicable to many other problems in many other settings, including settings in which users are not fully anonymous; we discuss only a few.

The present protocol is constructed to operate in a particular setting, in which stations have constant and identical traffic needs that can be efficiently satisfied in a round-robin fashion, and the protocol achieves efficiency in that setting. As we have already noted, the protocol can also operate, with similar efficiency, in the setting in which stations have constant but non-identical traffic needs, by the simple device of viewing a station with greater traffic needs as multiple stations. Similarly efficient protocols can be constructed to operate in more general settings. For instance, suppose that packet arrivals are "bursty" – perhaps with different arrival rates for different stations, so that some stations occasionally have many more packets to send than their average. Applying the protocol, as written, coordinates stations according to their average arrival rates, so that if station A has twice the average arrival rate of station B it will be allocated twice as many slots. (Note that the 802.11e wireless standard allocates slots according to initial requests of average rates but it does so in a centralized fashion; our protocol can accomplish the same allocation in a decentralized fashion.) This solves the coordination problem "on average" but it does leave open the problem of allocating additional slots on those occasions when the actual packet arrivals of station A are greater than the average arrival rate. One way to approach this problem is to view it as much like the problem of entry, already discussed (albeit briefly) in the paper. That is: when station A has many more packets than expected, it should behave "as if" the additional packets belonged to another station, and follow the entry protocol as discussed in the paper; this will have the effect of allocating additional slots to station A. However, this approach may not be good enough: if a large burst is an unusual occurrence, it would be wasteful to allocate these additional slots to station A on a permanent basis so in addition to an entry protocol there should be a corresponding exit protocol.[9] Alternatively, the protocol might be modified so that, during the coordination phase, stations signal the mean and variance (and perhaps other required traffic specification parameters) of their traffic needs for the protocol to allocate slots accordingly.

In Section V we have proposed a protocol for entry of secondary users. This is an appropriate protocol for environments in which primary users can be identified by their *physical signatures*, so that (re-)entry of primary users does not require signaling and coordination. For environments in which primary users cannot be identified by their physical signatures, the proposed entry protocol can be modified to allow efficient entry of primary users by directing primary and secondary users to create distinguishable *action signatures*.

There are of course other problems that our protocol does not address: How do secondary users detect primary users? How do secondary users decide which channel to attempt to access? How do secondary users detect less-used channels? To a considerable extent, these are individualistic problems, and a substantial amount of work has been done on these problems. Our work, which focuses on coordination problems, is complementary to this existing work. However, coordination may be relevant for some of these problems as well; for instance, if there are many channels, scanning of channels to

[8] We treat the switching problem as orthogonal to this paper; for an analysis of this problem, see [8] for instance.

[9] An exit protocol is discussed in [20]; that protocol could be modified to account for the possibility of errors.

detect those not occupied by primary users can be a demanding task, and could be made much more efficient if secondary users coordinate so that each secondary user only scans a subset of the channels. A similar issue arises with multi-user MAC power control protocols: which channel should each user use for transmission? It is well known that it is best for each user to concentrate power across a few channels (rather than spreading power across many channels), thereby reducing mutual interference. However this leaves open the coordination problem of which stations should use which channels and this is the kind of coordination problem that our methods can help solve.

REFERENCES

[1] 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*.

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

[3] Y. Chen, Q. Zhao and A. Swami, "Distributed spectrum sensing and access in cognitive radio networks with energy constraint," *IEEE Trans. Signal Processing*, vol. 57, no. 2, pp. 783-797, 2009.

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

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

[6] 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.

[7] 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.

[8] 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.

[9] J. Huang, R. Berry, and M. Honig, "Auction-based spectrum sharing," *ACM J. Mobile Net. and App.*, vol. 11, issue 3, Jun. 2006.

[10] 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.

[11] 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.

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

[13] 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.

[14] W. Zame, J. Xu and M. van der Schaar, "Winning the lottery: learning perfect coordination with minimal feedback," to appear in *IEEE J. Sel. Topics Sig. Process.*, 2013.

[15] 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.

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

[17] Y. Xu, J. Wang, Q. Wu, A. Anpalagan and Y.-D, Yao, "Opportunistic spectrum access in cognitive radio networks: Global optimization using local interaction games," *IEEE J. Sel. Topics Signal Process.*, vol. 6, no. 2, 2012, pp. 180-194.

[18] Y. Xu, J. Wang, Q. Wu, A. Anpalagan and Y.-D, Yao, "Opportunistic spectrum access in unknown dynamic environment: A game-theoretic stochastic learning solution," *IEEE Trans. Wireless Commun.*, vol. 11, no. 4, 2012, pp.1380-1391.

[19] M. Maskery, V. Krishnamurthy and Q. Zhao, "Decentralized dynamic spectrum access for cognitive radios: Cooperative design of a non-cooperative game," *IEEE Trans. Commun.*, vol. 57, no. 2, 2009.

[20] W. Zame, J. Xu and M. van der Schaar, "Cooperative Learning and Coordination for Cognitive Radio Networks," UCLA technical report, online at http://medianetlab.ee.ucla.edu/data/publications/CogRad.pdf.

APPENDIX: PROOFS AND FLOW CHARTS

Proof of Lemma 1 Consider a single station z. If the system is coordinated then z is silent in r_1+1 slots in each of ℓ digits, and its observation should be IDLE in every one of these slots. z fails the sync test if for at least one of these digits it observes BUSY in at least m_1 of these slots so the probability that z fails the sync test is at most $\ell B(m_1,(r_1+1);\varepsilon)$. Hence the probability that at least one of the N stations fails the sync test – which in this case means gets the wrong answer – is at most $N\ell B(m_1,(r_1+1);\varepsilon)$. □

Proof of Lemma 2 Consider a single station z. If the system is not coordinated then there is at least one digit in which z is silent and its observation should be BUSY in every one of (r_1+1) slots. z fails the sync test if it observes BUSY in at least m_1 of these slots so the probability that z fails the sync test in this digit is at least $B(m_1,(r_1+1);1-\varepsilon)$ and the probability that z passes the sync test in this digit at most $1-B(m_1,(r_1+1);1-\varepsilon)$, and the probability that z passes the sync test overall is no bigger. Hence the probability that at least one of the N stations passes the sync test – which in this case means gets the wrong answer – is at most $N[1-B(m_1,(r_1+1);1-\varepsilon)]$. □

The proofs of Lemmas 3, 4 are essentially identical and are omitted.

Proof of Theorem 1 (1) Assume there are no errors; we count the endogenous number of cycles. By definition and construction, the algorithm converges as soon as all stations pass the sync check. This requires N cycles of the Coordination Phase that end in WIN and $N-1$ cycles that end in HIT (for simplicity we treat this as N); all other cycles end in EMPTY or NOISE and are "wasted". WIN Cycles begin with a single active station, and that station is not randomizing; all other cycles begin with at least two active stations and all active stations are randomizing. Let C be the number of cycles that do begin with randomization. If a HIT cycle and a wasted cycle were equally likely, we would expect that to get N HIT cycles we will also need N wasted cycles, of which half will be EMPTY and half will be NOISE. (In fact a HIT cycle is more likely that a wasted cycle when the number of active stations is at least 3 but our estimation procedure does not take account of that fact.) So we estimate the probability that at least N of these C cycles are HIT cycles and at least $N/2$ of the remaining cycles are EMPTY cycles, which means that at most $C-3N/2$ are NOISE. A cycle ends in EMPTY if all active stations randomize and receive silent; a cycle ends in NOISE if all active stations randomize and receive transmit; other randomizing cycles end in HIT. Hence the probability that a randomizing cycle ends in HIT is at least .5 and conditional on a randomizing cycle *not* ending in HIT the probability that it ends in EMPTY is exactly .5. We now apply the sharp Chernoff bounds: 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 $e^{-2a^2/K}$. The probability we want is the probability of at least N HIT cycles *and* at least $N/2$ EMPTY cycles – which leaves at most $C-3N/2$ NOISE cycles. A little algebra shows that the probability of at least N HIT cycles in C cycles is at least $1-\exp[-(C-2N)^2/2C]$ and that the probability of at least $N/2$ EMPTY cycles in $C-N$ remaining cycles is at least $1-\exp[-(C-2N)^2/2(C-N)]$. Hence the probability of at least N HIT cycles *and* at least $N/2$ EMPTY

cycles is at least $1-2\exp[-(C-2N)^2/2(C-N)]$. If there are N HIT cycles and $N/2$ EMPTY cycles and the remaining $C-3N/2$ cycles are NOISE then the total number of slots will be $2N+N/2+3(C-3N/2)=3C-2N$. Since we have D^* slots to use (before the protocol enters the Synchronization Phase) and $3N$ of these slots will be used up by N WIN cycles, we need take $3C-2N=D^*-3N$ or $3C=D^*-N$. Hence $C-2N=(D^*-7N)/3$ and $C-N=(D^*-4N)/3$ so $-(C-2N)^2/2(C-N)=-2(D^*-7N)^2/3(D^*-4N)$; substituting gives the desired expression.

(2) The probability of one station making an error in one slot is at most ϵ so the probability that at least of N stations makes an error in at least on of D^* slots is at most $ND^*\varepsilon$ and the probability of no error is at least $1-ND^*\varepsilon$.

(3) The probability of convergence in one round is at least the product of the probability of convergence in one round conditional on no errors times the probability of no errors in the first D^* slots times the probability of getting the right answer in the Synchronization Phase.

(4) If convergence fails in the first round *and* all stations agree that convergence has failed in the first round (get the right answer), then the protocol starts again – but if some station believes convergence has succeeded (gets the wrong answer) then failure is permanent. Hence the probability of convergence in the second round is the probability of failure to coordinate in the first round times the probability that failure is detected times the probability of success in the second round, and so forth for any number of rounds. This gives an exact formula for the probability of convergence in at most R rounds. We do not have exact values for the probability of coordination in each round conditional on no errors, for the probability of no errors, and for the probability of getting the wrong answer in the Synchronization phase, but we do have lower bounds for all of these probabilities, and it is easy to check that substituting the lower bounds for these probabilities into the exact formula for convergence in at most R rounds gives a lower bound for the probability of convergence; this is the result asserted. □

Proof of Theorem 2 The proof is identical to that of Theorem 1, except that WIN cycles, HIT, NOISE slots all take 4 slots. □

William Zame is Distinguished Professor in the Departments of Economics and of Mathematics at UCLA and past Director of the California Social Sciences Experimental Laboratory. He is the author/co-author of more than 80 journal articles in pure mathematics, economic theory, finance, experimental economics, and economics and engineering. He is a Fellow of the Econometric Society and was a Fellow of the John Simon Guggenheim Memorial Foundation for 2004-2005.

Jie Xu received the B.S. and M.S. degrees in Electronic Engineering from Tsinghua University, Beijing, China, in 2008 and 2010, respectively. He is currently a Ph.D. student with the Electrical Engineering Department, UCLA. His primary research interests include wireless communications, cooperative communications, game theory and learning in multi-agent networks.

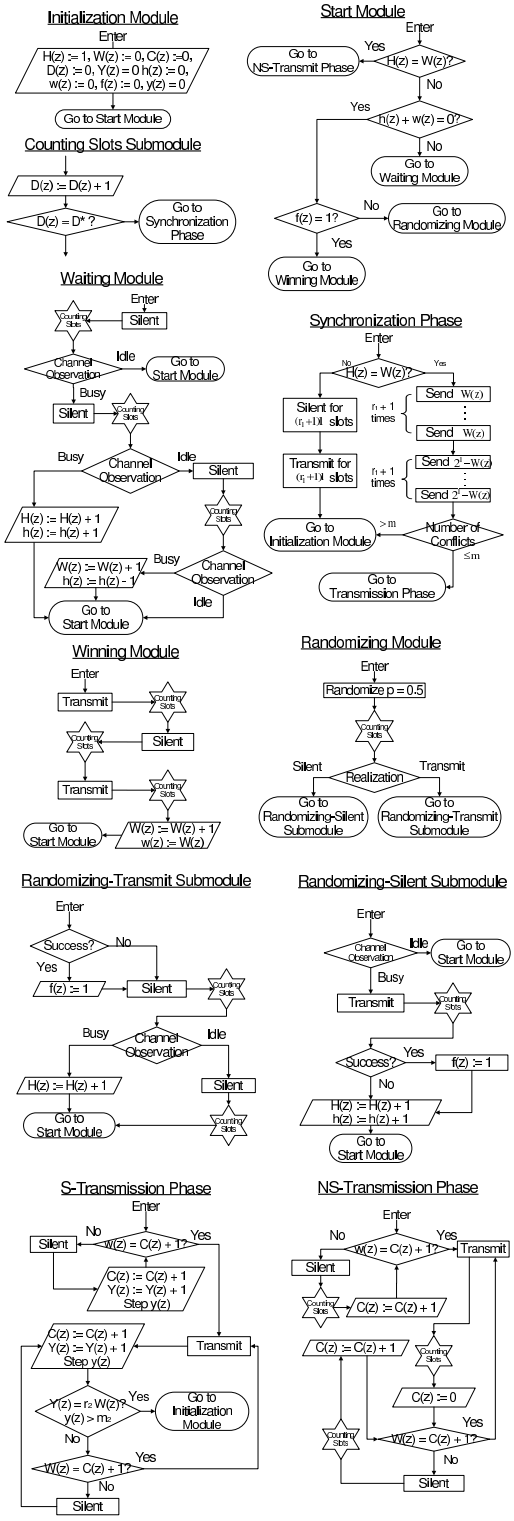

Fig. 6. Flow Charts for the Basic Protocol

Mihaela van der Schaar (F'10) is currently a Chancellor's Professor in Electrical Engineering Department, UCLA. Her research interests include communications, engineering economics and game theory, strategic design, online reputation and social media, dynamic multi-user networks and system designs.