Fast Rendezvous for Multiple Clients for Cognitive Radios Using Coordinated Channel Hopping

Rohan Gandhi, Chih-Chun Wang, and Y. Charlie Hu
Purdue University

***Abstract*—A primary challenge in exploiting Cognitive Radio Networks (CRNs), known as the rendezvous problem, is for the users to find each other in the dynamic open spectrum. We study blind rendezvous, where users search for each other without any infrastructural aid. Previous work in this area have focused on efficient blind rendezvous algorithms for two users but the solution for multiple users is still far from optimal. In particular, when two users encounter, one user inherits the other's hopping sequence but the sequence is never *shortened* or *split* among the encountering users. We denote this class of algorithms as uncoordinated channel hopping algorithms. In this paper, we introduce a new class of distributed algorithms for multi-user blind rendezvous, called Coordinated Channel Hopping (CCH), where users adjust, or coordinate, the sequence of channels being hopped as they rendezvous pairwise. Compared to existing rendezvous algorithms, our algorithms achieve 80% lower Time To Rendezvous (TTR) in case of multiple users.**

I. Introduction

The wireless spectrum today consists of licensed and unlicensed spectrum. Due to the evolution of wireless devices and WiFi that operate in the unlicensed spectrum, the unlicensed spectrum is over-crowded. In contrast, the licensed spectrum is underutilized. Cognitive radio networking (CRN) is a promising way to improve the spectrum usage. The users in the CRN are referred as *cognitive users* (CUs) that operate in a spectrum licensed to the owners of the spectrum (referred as *primary users* or PUs). The CUs use the cognitive radio to identify unused licensed spectrum (*i.e.,* idle channels) and hop on the idle channels without causing any interference to the PUs.

A CRN potentially consists of many users. At any given time, two or more users may wish to communicate with each other, for which they need to first establish a common communication channel by somehow tuning in to that channel at the same instance of time, also known as achieving *"rendezvous"*. Designing efficient algorithms for multiple users in a CRN to quickly achieve rendezvous has been under intensive study in the past few years.

The rendezvous problem in a CRN is generally framed in two scenarios in terms of channel availability per user. In **symmetric** scenarios, the set of channels available are the same for all the users. In **asymmetric** scenarios, different set of channels are available to different users. The major challenge in blind rendezvous is to provide the guaranteed rendezvous in asymmetric scenarios, as in symmetric scenarios, all users can just linger on the lowest channel to reach rendezvous.

To date, most of the rendezvous algorithms use either a central controller or a dedicated channel to aid the users achieve rendezvous (*e.g.,* [11], [2]). The dedicated channels are referred as *Common Control Channels* (CCC). Though these approaches are simple to implement, they suffer from several drawbacks: (1) Establishing CCC is not always feasible because any preselected CCC may be in use by the PUs; (2) The central control and the CCC do not scale well as the number of users increases; (3) The central controller presents a central point of failure.

To overcome the above drawbacks, distributed, *blind rendezvous* algorithms which do not require a central controller or a CCC have been proposed. A major class of these algorithms (*e.g.,* [8] [14]) use the channel hopping (CH) technique, where the users hop the idle channels using a particular hopping sequence so that all of them arrive at the same channel in the same time-slot.

Jump-Stay [8] (JS) is the state-of-the-art distributed algorithm that provides guaranteed blind rendezvous in asymmetric scenarios. JS achieves the guarantee via a novel technique that combines channel hopping with a *stay* stage during which each user stays on a particular channel for a number of time-slots. In the multi-user case, each user performs jump-stay as before, except at each pairwise encounter one user copies the jump-stay parameters from the other (following an implicit ordering of the users). After enough pairwise encounters, all users will have the same jump-stay parameters and hence hop along the same channel sequence, reaching rendezvous. We thus refer to algorithms like JS as Uncoordinated Channel Hopping (UCH) algorithms as the sequence of channels being hopped though propagated is never *shortened* or *split* among the users.

In this paper, we introduce a new class of distributed algorithms for multi-user blind rendezvous, called Coordinated Channel Hopping (CCH), where users adjust, or coordinate, the sequence of channels being hopped as they rendezvous pairwise. We propose two CCH algorithms, Iterative Intersection Hopping (IIH) and Divide and Conquer Hopping (DCH). The two CCH algorithms explore two directions to accelerate the rendezvous between a new user and existing users which typically have already achieved rendezvous. IIH focuses on having the new user quickly meeting the first one of the existing users, while DCH focuses on quickly spreading the information about the new user among the existing users. How to simultaneously optimize both remains an open problem. Our new algorithms guarantee rendezvous. Simulation studies show they reduce the time to rendezvous (TTR) by 80% on average compared to Jump-Stay.

The rest of the paper is organized as follows. Section II discusses the related work. Section III presents the system model

and notations used in the paper. Section IV and Section V describe our rendezvous algorithms in two-user and multi-user scenarios, respectively. Section VI presents performance comparison, and Section VII concludes the paper.

II. Related Work

The existing rendezvous algorithms can be categories into: centralized or decentralized.

a) Centralized algorithms: In the centralized algorithms, a central controller or a base station assists in the rendezvous process, where the users can contact the central controller and the central controller can direct the users to the same channel. The central controller also periodically scans the spectrum, in order to identify the idle channels. The challenge in the centralized algorithms is to enable the users find the channels on which the central controller is operating.

The centralized algorithms can be classified into two classes based on whether they require pre-selected CCC or not. In the centralized algorithms that require pre-selected CCC [11] [2], users contact the central controller using pre-selected channels. The pre-selected channels are the channels that are reserved for the control communication with the central controller and these channels are well known to the users.

Using a pre-selected CCC has several drawbacks that limit scalability, including: 1) Allocation and maintenance of the CCC can be costly as the CRN operates in presence of one or more PU, 2) Congestion on the CCC as a result of increased traffic when the number of users increases.

The other class of the centralized algorithms do not require the pre-assigned CCC. One implementation for this class of the rendezvous algorithms is explained in [6] that uses exhaustive mechanism for rendezvous.

Though easy to implement, centralized algorithms face problems that include: 1) Failure of the central controller, 2) Centralized controller may become overloaded and thus become a bottleneck in case of multiple users.

The decentralized algorithms on the other hand, do not suffer from the above-mentioned drawbacks, but they are not simple to implement too. Using decentralized algorithms, it is challenging to provide guaranteed rendezvous in finite time. Similar to the centralized algorithms, the decentralized algorithms can further be classified in to 2 classes depending on whether they require the CCC or not.

b) Decentralized algorithms that require CCC: [4] [10] implement the decentralized algorithms that require the CCC to be given in advance. However, given the uncertain nature of the availability of channels in the CRN, the availability of the CCC is not always guaranteed and therefore, these algorithms requiring global CCC are not always feasible.

The algorithms in [3] [5] [7] [18] take advantage of the common channels within the vicinities of the users rather than global CCC to contact their neighbors. These algorithms do not suffer from the disadvantages of the global CCC, but add considerable overhead to establish and maintain their CCCs.

c) Decentralized algorithms that does not require CCC: As the CCC based decentralized algorithms have their limitations, decentralized algorithms that do not require any CCC are becoming popular among the researchers. These algorithms are also known as the blind rendezvous algorithms. In the blind rendezvous algorithms, each user only knows the channels available to itself in the beginning. Rendezvous can be achieved on any one of these available channels.

One way to achieve blind rendezvous is by using *channel hopping* (CH) techniques. In CH techniques, users hop on the available channels in a particular way to achieve rendezvous across all the users in finite time.

The earlier blind rendezvous algorithms such as [14] provides 3 blind rendezvous rendezvous algorithms: Generated Orthogonal Sequence (GOS), Modular Clock (MC) and Modified Modular Clock (MMC). GOS generates random CH sequence. MC and MMC work on the prime number and modulo operations. These algorithms select prime numbers depending on the number of channels available and then pick up the *jump*-step less than the prime number. The hopping sequence is generated based on the jump-step and the prime number.

GOS and MC algorithms work in the symmetric model for the two users. However, GOS and MC do not provide the guaranteed rendezvous in finite time. MMC was proposed to work in the asymmetrical model where both users observe different channels available. Again, MMC does not provide any guarantee for the finite time rendezvous.

Jump-Stay algorithm presented in the [8], [9] builds on top of the MC. The drawback of the MC algorithm was that the rendezvous is not guaranteed if the jump-step is the same for both users. To resolve this problem Jump-Stay algorithm introduced a *stay* stage where the users stay on a particular channel for time-slots equal to the prime number. The Jump-Stay algorithm thus provides the guaranteed rendezvous for two users as well as for the multiple users.

As achieving rendezvous is a critical problem in the CRN, there had been proposals from different prospects. Due to lack of space, we cannot cite all of the proposals and mention about the broad class they represent.

One class of the decentralized rendezvous systems is that doesn't use CH mechanisms. [1], [6] use leader selection algorithms to set up an infrastructure to aid the rendezvous process. Similarly, [12] proposes a grid based method that promises rendezvous with high probability.

Also, there had been proposals in the wireless sensor network [16], where the rendezvous is achieved using a channel scanning directed by the link quality. Also, there had been proposals in spectrum management such as [15], [13] which mainly focus on the smooth hand-offs to reduce communication disruption.

Finally, Though most of the work in this area had been through simulations, a very few implementations have been done such as [17].

III. System Model

Our system for the cognitive radio consist of 3 components: 1) Maximum number of channels available (N), 2) Number of users (L), 3) Symmetry between the channels available to individual users.

We consider a CRN consisting of L users. $L \geq 2$ and all the users are equipped with the radio. The users in the CRN operate in a spectrum that is licensed to one or more PUs. This licensed spectrum is divided into N ($N \geq 1$) non-overlapping channels that can potentially be used by the CRN users and therefore, N denotes maximum number of channels available to any user. The channels are uniquely indexed by 1,2,..,N. This set of potential available channels is denoted by $\mathcal{C} = \{c_1, c_2, ..., c_N\}$, where c_i denotes the i^{th} available channel.

$\mathcal{C}_i$ denotes the set of available channels to user *i*. Note that $\mathcal{C}_i$ can be different than $\mathcal{C}$ as there can be some channels in $\mathcal{C}$ that are not available to user *i*. These channels can be occupied by the PUs that are operating in close vicinity to the user *i*. We call set $\mathcal{C}_i$ the **working set** for user *i*. For example, let's consider a scenario with $N = 5$ and $L = 2$. The available channels are $\{c_1, c_2, c_3, c_4, c_5\}$. It may happen that channels $\{c_1, c_2, c_4\}$ are available to user L_1 and channels $\{c_3, c_4\}$ are available to user L_2. Therefore $\mathcal{C}_1 = \{c_1, c_2, c_4\}$ and $\mathcal{C}_2 = \{c_3, c_4\}$. As $\mathcal{C}$ consists of all the channels available $\mathcal{C}_i \subseteq \mathcal{C}$.

Finally, there must be at least one channel that is common across all the users for the successful rendezvous. This set of common channels across all the users is denoted by $\overline{\mathcal{C}}$ and $\overline{\mathcal{C}} = \bigcap_i \mathcal{C}_i$. It should be noted that by definition, the working set for each user contain the set $\overline{\mathcal{C}}$. *i.e.,* $\overline{\mathcal{C}} \subseteq \mathcal{C}_i$. We denote number of channels in set $\overline{\mathcal{C}}$ by G. Table I summarizes the key notations used in this paper.

The rendezvous problem can be categorized into 2 different scenarios based on the number of the users in CRN:

1) 2-user: A CRN where there are 2 users trying to achieve rendezvous *i.e.,* arrive at the same channel at the same time.

2) Multi-user: A CRN where multiple users ($L > 2$) are trying to rendezvous *i.e.,* all the users are trying to arrive at the same channel in the same time-slot.

For both cases we consider the following two models.

1) Symmetric model: In this model all the channels are available to all the users. *i.e.,* $\mathcal{C} = \overline{\mathcal{C}} = \mathcal{C}_i$ for $1 \leq i \leq L$.

2) Asymmetric model: In the asymmetric model channels available to both users can be different. Therefore it might be possible to have $\mathcal{C}_i \neq \mathcal{C}_j$ for $1 \leq i, j \leq L$.

In any CRN, users can start at any time. Moreover, in the blind rendezvous algorithms, users do not have any information about other users when they arrive on the network. This information includes available channels, the hopping sequence, the starting time. The only information available to each user is its own working set and the maximum available channels N. With this requirement, we define the **rendezvous problem** as: Given that the CRN consists of L, $L \geq 2$ users starting in asynchronous manner and only with the information about their own working set, the problem is to form an algorithm to guarantee that all the users hop the channels without causing any interference to the PUs, such that they arrive at the same channel in the same time-slot. Note that, similar to previous work [8] [14], we focus on the CH algorithms in a time-slotted system. In practice, in addition to the CH algorithm, other processes such as beaconing and handshaking are required. These processes also determine the duration of each time-slot.

The performance of different algorithms can be compared based on the *Time to Rendezvous (TTR)* and the *guarantee*. TTR is defined as the number of time-slots elapsed before the rendezvous happens. For the 2-user scenario, the TTR is measured from the time slot when the 2^{nd} user starts, assuming that the 1^{st} user starts in time-slot 0.

IV. 2-User Scenario

In the asymmetric model, both users can have different working sets, *i.e.,* $\mathcal{C}_1 \neq \mathcal{C}_2$. Unlike the symmetric case, in this scenario the users can not simply hop on the smallest channel as the channel may not be available for both users or both users may have different smallest channels. Therefore, each user needs to hop the channels available to it, such that eventually it can achieve rendezvous with the other user. We use the *Hopping-Sequence* (HS) function to generate the hopping sequence.

*1) **HS function**:* This function generates a hopping sequence for each round. One round consists of $(2 \cdot P^2 + 2 \cdot P)$ time-slots, where P denotes a prime number ($P > N$). $2 \cdot P^2$ time-slots constitute the *"jump"* stage, which is followed by the remaining $2 \cdot P$ time-slots constituting the *"stay"* stage. In the *jump* stage, users can hop on different channels in different time slots, whereas in the *stay* stage users hop on a particular single channel. The *jump* stage is further divided into *P inner-rounds* of $2 \cdot P$ time-slots each. The intuition for this design goes to guarantee rendezvous as we explain later in this section.

HS function generates hopping sequence based on the parameters shown in table II. The HS function uses 2 random sequences, which are denoted by L_A and L_B. Based on the value of P, L_B is generated as a randomly ordered list from 0 to P-1. Both users must have the same L_B to guarantee rendezvous, which is not hard as both the users use the same N and P. Unlike L_B, sequence L_A is generated based on the working set of that particular user. As both users can have different working sets, L_A can be different.

We use *r* and *x* such that the users hop on the channel generated by the equation $c = (r \cdot x + t)\%P$, where *c* is the channel index, *t* is the time. The HS algorithm is run by each user independently and therefore *r* and *x* can be different for both users.

Formally the channel hopping mechanism is described in the HS function shown in figure 1. In line 3, t_1 specifies the time with respect to each round. In line 4, *r* specifies the *rate*. We also use *r* to select the channel to hop during the *stay* stage. *r* is unchanged throughput the round but changes in every round. In the *jump* stage, *y* determines the time with respect to the inner-round, whereas *x* determines the offset. *z*, which represents the channel index is calculated in line 8

TABLE I
LIST OF KEY NOTATIONS.

Term	Definition
N	Total number of available channels
L	Total number of cognitive users
$\mathcal{C}_i$	Set of channels available to i-th user
$\mathcal{C}$	Set of all channels. $\mathcal{C} = \bigcup_i \mathcal{C}_i$
$\overline{\mathcal{C}}$	Set of common channels $\mathcal{C} = \bigcap_i \mathcal{C}_i$
G	Number of channels in $\overline{\mathcal{C}}$

TABLE II
LIST OF PARAMETERS TO GENERATE HOPPING SEQUENCE.

Term	Definition
P	Smallest prime number such that $P > N$
L_A	Set of current channels available
L_B	Random sequence between 0 to P-1
r	Rate of hopping sequence
y	Time relative to the inner-round
x	Offset of entry in the L_B
z	Channel index

§ HS FUNCTION

1. **Input**: N, P, L_B, L_A, t
2. **Output**: Channel index c
3. $t_1 \leftarrow$ t mod $(2P^2 + 2P)$
4. $r = \left\lfloor \frac{t}{2P^2+2P} \right\rfloor \%P$
5. **if** $t_1 < 2P^2$ **then**
6. $\quad y \leftarrow t_1$ mod 2P
7. $\quad$ x = $\lfloor \frac{t_1}{2P} \rfloor$-th entry of L_B
8. $z \leftarrow (r \cdot x + y)\%P + 1$
9. **else**
10. $\quad z = r + 1$
11. **end if**
12. **if** z is in L_A
13. $\quad c = z$.
14. **else**
15. $\quad$ c = random channel in L_A
16. **end if**

Fig. 1. Pseudo-code for constructing the hopping sequence.

based on the values of *x* and *y*. We use P-modulo to limit the value of z under P. If the channel indexed by z is in L_A, then z is returned. Otherwise, a randomly chosen channel index is returned as shown on line 15. In the *stay* stage, channel index indicated by *r* is returned.

A. Rendezvous Guarantee

The rendezvous is guaranteed using HS function because of the 2 main design choices:

- Use of $P > N$, so that all the N available channels are hopped in P time-slots.
- hopping channels according to $c = (r \cdot x + t)\%P$.

Firstly, we prove on why the N available channels are hopped in the P time-slots.

Corollary 1: All N channels are hopped in the P time-slots.

Proof: t increments for every time-slot, so are t_1 and y. Moreover, in each jump-stage, both r and x remain unchanged. Therefore during a duration of P time-slots in any jump-stage, the user will hop on channel $(r \cdot x + y)\%P + 1$, according to line 8 of figure 1. When y increments by 1 to P, all channels will be hopped at least once during these P time-slots. ■

Based on this property we now prove the rendezvous guarantee for the 2 users.

Theorem 1: The rendezvous is guaranteed for the 2 users in the asymmetric model.

Proof: We consider the worst case in which there is only one common channel, whose index is denoted by c^*. The two users are denoted by user-1 and user-2. With only one channel in common, rendezvous will be achieved when both users hop on channel c^* in the same time-slot. Without loss of generality, assume user-1 starts before user-2. Therefore when user-2 starts its "stay-on-c^*" stage (there may be some "stay-on-c" stages with $c \neq c^*$ before user-2 decides to stay on c^* in a particular round), user-1 has already started.

Let l denote the number of time-slots for which the "stay-on-c^*" stage of user-2 and overlaps with the *jump* stage of user-1. Two cases can occur depending on the value of l.

Case 1: $l \geq P$ as depicted in Fig. 2(a) and 2(b). Since the overlap is $\geq P$, from Corollary 1, user-1 will hop all N channels, including c^*, during the l overlapped time-slots. Rendezvous will be achieved when user-1 hops on c^* since user-2 is in its "stay-on-c^*" stage.

Case 2: $l < P$ as depicted in Fig. 2(c) and 2(d). There are 2 sub-cases. Case 2.1: In its current round, user-1 also hops on channel c^* in its *stay* stage. This scenario can occur when the r value chosen in line 4 of Fig. 1 is the same for both users (in their current rounds). Since both users hop on channel c^* in their stay stages and their stay stages overlap by $2P - l \geq 1$ time-slots, rendezvous is guaranteed.

Case 2.2: In its current round, user-1 hops on channel $c \neq c^*$ in its *stay* stage. For this case, rendezvous will occur during the *jump-jump* phase of both users due to the following reasons.

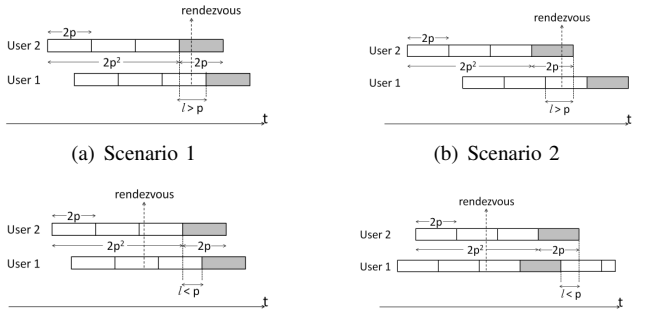
From Figs. 2(c) and 2(d), it can be seen that for user-2, all P jump stages before its "stay-on-c^*" stage are heavily overlapped (with overlap being $2P - l \geq P$) with the P jump stages of user-1 before its "stay-on-c" stage. Let δ denote how many times slots has user-1 arrived earlier than user-2. Since $c^* \neq c$, between 0 and $P - 1$ there exists a unique number Δ satisfying $(c^* - c)\Delta \equiv \delta \mod P$. We claim that both users hop on the common channel c^* during the overlapped portion of the jump stage when the x value in that jump stage is Δ.

The reason is as follows. Let $t^{(2)}$ denote the time-slot when user-2 hops on c^* in the jump stage when the x value is Δ. By Line 8, we must have $c^* = ((c^* - 1)\Delta + t^{(2)})\%P + 1$ (since for user-2, c^* is the stay stage and thus $r = c^* - 1$). Also the timer at user-1 must satisfy $t^{(1)} = t^{(2)} + \delta$. Therefore, at the same time user-1 hops on

$$\begin{aligned}&((c-1)\Delta + t^{(1)})\%P + 1 = ((c-1)\Delta + t^{(2)} + \delta)\%P + 1 \\ &\overset{(a)}{=} ((c^* - 1)\Delta + t^{(2)})\%P + 1 = c^*.\end{aligned}$$

where (a) is due to that Δ satisfies $(c^* - c)\Delta \equiv \delta \mod P$. As a result both users hop on the common channel c^* at the same time. Rendezvous is thus guaranteed. ■

*1) **Maximum time to rendezvous**:* Complexity analysis is often performed in terms of Maximum time to rendezvous. In the worst case, there is only one channel common between the 2 users and the users achieve rendezvous in their *stay* stage as shown in case 2.1. It can take up to P rounds for the user-1 to hop on c^*. When the user-1 is hopping on the channel c^*,



(a) Scenario 1 (b) Scenario 2

(c) Scenario 3 (d) Scenario 4

Fig. 2. Different scenarios in 2-user case. The stay stage for both the users is shown as a shaded region. User-2 hops on the channel c^* in its stay stage.

§ HS ALGORITHM

1. **Input**: N
2. Find smallest prime number P, such that $P > N$
3. Calculate L_B based on P and L_A based on the working set
4. **while** (not rendezvous)
5. c = HS(N, P, L_B, L_A, t)
6. Attempt rendezvous on channel c
7. **end while**

Fig. 3. HS algorithm to achieve rendezvous.

user-2 can take up to $2 \cdot P^2 + 2 \cdot P$ time-slots to rendezvous. This way the maximum TTR possible is: $P \cdot (2 \cdot P^2 + 2 \cdot P)$.

*2) **Sensitivity to dynamic PU activity**:* Cognitive users operate in the frequency band currently unoccupied by the primary users. However, the PU activity can be dynamic wherein a PU can occupy or unoccupy a channel any time. All the CUs update their working set periodically to account for the dynamic PU activity. The frequency of the updates depends on the frequency of the PU activity.

Our algorithm is unaffected by the PU activity as long as $\overline{\mathcal{C}} \neq \phi$ and the rendezvous is still guaranteed within $P \cdot (2 \cdot P^2 + 2 \cdot P)$ time-slots .

*3) **HS algorithm**:* HS algorithm uses HS function to generate the hopping sequence as shown in figure 3. In line 2, smallest prime number P is found such that $P > N$. Then the user formulates the sequences L_A and L_B. In line 5, HS function is called, which returns the channel index. Line 5 is repeated until the rendezvous is achieved.

V. MULTI-USER SCENARIO

In the multi-user scenario, the number of users in the CRN is greater than 2. In this section we describe our algorithms to achieve rendezvous among all the users for asymmetric model.

The rendezvous problem in the asymmetric case can be analyzed by analyzing the expected TTR in a k-user scenario where a new user L_{k+1} arrives on the network when previous *k* users have achieved rendezvous. This result can be approximated to the worst case when the users arrive asynchronously. As we explain in this section, the rendezvous can occur in the next time-slot once all *k* users have working set information about the L_{k+1} user and vice-a-versa.

We divide the timeline for the (k+1) users to achieve rendezvous in 2 phases:

1) L_{k+1} user achieves rendezvous with any one of the *k* users. At this point, the user L_{k+1} gets the information about all the *k* users.
2) The working set of the L_{k+1} user is spread among all the *k* users.

Therefore, the expected TTR = Time to finish phase (1) + Time to finish phase (2).

In the multi-user scenarios, the Jump-Stay algorithm makes the users generate the same hopping sequence as they pairwise rendezvous. With the identical hoping sequence, users hop on the same channels[1], which is equivalent to only one user hopping the channels. Jump-Stay algorithm does not take advantage of two observations:

- Hopping on the common channels ($\overline{\mathcal{C}}$). As the working sets for all the users contain $\overline{\mathcal{C}}$, hopping channels from $\overline{\mathcal{C}}$ will increase the number of times the users achieve rendezvous and thus, would reduce the TTR.
- 2 (or more) users after achieving pairwise rendezvous, can hop independently on different channels in the same time-slot to accelerate the process to achieve rendezvous among all the users.

Based on these observations we propose two algorithms to achieve rendezvous among multiple users to reduce the completion time for phase (1) and phase (2):

- Iterative Intersection Hopping (IIH).
- Divide and Conquer Hopping (DCH).

A. Iterative Intersection Hopping

The IIH is aimed towards reducing the time for the phase (2) by taking advantage of observation to hop on the common channels $\overline{\mathcal{C}}$. In a k-user scenario, if any of the *k* users has achieved rendezvous with the $(k+1)^{th}$ user, the information can be spread among the *k* users quickly if they are hopping on the common channels.

However, finding $\overline{\mathcal{C}}$ is not trivial in the blind rendezvous algorithms as the users only know about the channels in their working set and have no information about the working set of other users in the beginning. IIH handles this challenge by changing the working set of the users to their mutual common channels whenever they rendezvous.

The algorithm has the following two main building blocks:

*1) **Change in working set**:* In this algorithm, when the two users achieve rendezvous, the users can easily exchange their working sets on the same channel where the rendezvous was achieved. If user L_i and user L_j achieve rendezvous, new working set for users L_i and L_j ($\mathcal{C}_i$ and $\mathcal{C}_j$) becomes $\mathcal{C}_i \cap \mathcal{C}_j$.

As a user achieves rendezvous with more and more different users, the working set keeps reducing and finally it will be reduced to set $\overline{\mathcal{C}}$. This is because, when user L_i achieves rendezvous with all other users, the working set for L_i become $\overline{\mathcal{C}_i} = \bigcap_i \mathcal{C}_i$. By definition, $\overline{\mathcal{C}} = \bigcap_i \mathcal{C}_i$. Therefore $\overline{\mathcal{C}_i} = \overline{\mathcal{C}}$.

[1]If the channel is not available to a user, then that user hops on a random channel in that time-slot. However, if a channel is available to both users, then they guaranteed hop that channel in the same time-slot.

§ ITERATIVE INTERSECTION HOPPING

1. **Input**: N
2. Find smallest prime number P, such that $P > N$
3. Calculate L_B based on P
4. Calculate L_A based on the working set
5. **while** (not terminated)
6. $c = HS(N, P, L_B, L_A, t)$
7. **if** (successful rendezvous)
8. Exchange common channels
9. $\mathcal{C}_i = \mathcal{C}_i \bigcap \mathcal{C}_j$
10. **if** (change in working set)
11. Recalculate L_A
12. **end if**
13. Update the user count
14. **end if**
15. **end while**
16. c = smallest channel in $\overline{C}$

Fig. 4. Pseudo-code for IIH.

Once the working set for all the users become $\overline{\mathcal{C}}$, the rendezvous problem for the asymmetric model converges to the rendezvous problem for the symmetric model and thus the users can simply hop on the smallest channel in $\overline{\mathcal{C}}$.

*2) **User count and information**:* When the two users rendezvous, they exchange the information about the other users they have previously achieved rendezvous. This information is used to terminate the algorithm faster. When user L_i achieves rendezvous with user L_j, the working set for both users become $\mathcal{C}_i \cap \mathcal{C}_j$. Later if user L_i achieves rendezvous with user L_k, the working set for user L_k (and L_i) becomes $\mathcal{C}_i \cap \mathcal{C}_j \cap \mathcal{C}_k$. As L_k already has its working set reduced to the common channels between L_i, L_j and L_k, it does not need to achieve rendezvous with user L_j before termination, which eliminates the need to achieve rendezvous between each pair of users and importanly, accelerates the rendezvous process.

*3) **Algorithm**:* The algorithm is formally showed in the figure 4. In this algorithm, the users hop the channels according to the HS algorithm as shown on line 6. Whenever a pairwise rendezvous happens, both users exchange the channels available to them to formulate the new working set and recalculate the L_A sequence, as shown on line 11. Also, the users update their user count as shown in line 13.

*4) **Expected TTR**:* To find the expected TTR in the multi-user scenario, we first calculate the expected TTR for the 2 clients in the following Corollary.

Corollary 2: The expected time to achieve rendezvous for 2 users is $\frac{N_1 \cdot N_2}{\overline{\mathcal{C}}}$, where N_1 and N_2 are the number of channels in the working set of user L_1 and L_2 respectively.

Proof: For the two users to achieve rendezvous, they need to hop the same channel in the same time-slot. The channel on which the users have achieved rendezvous is denoted by c. If L_1 is hopping on N_1 channels, the probability of selecting channel c is $\frac{1}{N_1}$. Similarly, for the user L_2, the probability to select channel c is $\frac{1}{N_2}$. Therefore, the probability for both users to arrive at the channel c is $\frac{1}{N_1 \cdot N_2}$. As there are $\overline{\mathcal{C}}$ common channels between both users, the probability to achieve rendezvous in any round is $\frac{\overline{\mathcal{C}}}{N_1 \cdot N_2}$. Therefore, the expected time to achieve rendezvous is $\frac{N_1 \cdot N_2}{\overline{\mathcal{C}}}$. Note that, this expected time to rendezvous applies only when both the users are in the *jump* stage. ■

Using Corollary 2, we now show the expected TTR (ETTR) for the IIH algorithm in a scenario where $(k+1)^{th}$ user joins when the previous k users have already achieved rendezvous.

Corollary 3: The expected time for the $(k+1)^{th}$ user to achieve rendezvous when previous k users have achieved rendezvous is $\frac{N_{k+1}}{G_{k+1}} + G_k$, where G_k and G_{k+1} denote number of common channels between the k and (k+1) users respectively.

Proof: We calculate the ETTR in phase 1 and phase 2.

a) Phase 1: As k users have achieved rendezvous, the working set of all the users is equal to $\overline{\mathcal{C}_k}$, where $\overline{\mathcal{C}_k} = \bigcap_i \mathcal{C}_i$ for $i = 1, .., k$. Let $G_k = \left|\overline{\mathcal{C}_k}\right|$, denote the number of channels in the set $\overline{\mathcal{C}_k}$. Similarly, $\overline{\mathcal{C}_{k+1}} = \bigcap_i \mathcal{C}_i$ for $i = 1, .., k+1$ and $G_{k+1} = \left|\overline{\mathcal{C}_{k+1}}\right|$.

From Corollary 2, the probability for one of the k users to achieve rendezvous with the $(k+1)^{th}$ user is $\frac{G_{k+1}}{G_k \cdot N_{k+1}}$. Therefore the probability for any of the k users to achieve rendezvous with the $(k+1)^{th}$ user is $1 - (1 - \frac{G_{k+1}}{G_k \cdot N_{k+1}})^k$, approximated to $p = 1 - (1 - \frac{k \cdot G_{k+1}}{G_k \cdot N_{k+1}}) = \frac{k \cdot G_{k+1}}{G_k \cdot N_{k+1}}$ as $N_{k+1} > G_{k+1}$. Therefore, ETTR(phase 1) = $\frac{1}{p} = \frac{G_k \cdot N_{k+1}}{k \cdot G_{k+1}}$.

b) Phase 2: The working set for the $(k+1)^{th}$ user and one of the k users is $\overline{\mathcal{C}_{k+1}}$ at this point. The working set for the other users is unchanged and it is $\overline{\mathcal{C}_k}$. Therefore, the expected time to achieve the rendezvous with one of the remaining users is $\frac{G_{k+1} \cdot G_k}{G_{k+1}}$ according to Corollary 2. As the expected TTR for each user is independent, the ETTR(phase 2) = $\frac{G_{k+1} \cdot G_k}{G_{k+1}} = G_k$ and thus,

ETTR(total) = ETTR(phase 1) + ETTR(phase 2) = $\frac{G_k \cdot N_{k+1}}{k \cdot G_{k+1}} + G_k$.

However, as all the k users have $\overline{\mathcal{C}_k}$ as the working set and if $(k > G_k)$, then there is at least one channel in G_k being hopped by at least 2 users. If two (or more) users out of k users are hopping the same channel, then it is equivalent to one user hopping the channel. Therefore, the TTR becomes $\frac{G_k \cdot N_{k+1}}{\overline{k} \cdot G_{k+1}} + G_k$, with $\overline{k} \leq G_k$ because G_k number of channels are being hopped collectively. Therefore ETTR(total) becomes, $\frac{G_k \cdot N_{k+1}}{G_k \cdot G_{k+1}} + G_k = \frac{N_{k+1}}{G_{k+1}} + G_k$. ■

The Jump-Stay algorithm, on the other hand, required $\frac{N_k \cdot N_{k+1}}{G_{k+1}}$ time-slots because in the Jump-Stay algorithms all the k users are hopping using the same hopping sequence, which is equivalent to one user hopping the same sequence. We show this behaviour in figure 11(a) in our evaluation.

B. Divide and Conquer Hopping

"Divide and Conquer Hopping" (DCH) has a different principle than IIH. However, it borrows some of the building block elements from IIH such as hopping on the common channels. DCH is aimed towards improving the completion time for phase (1) and phase (2).

The idea behind DCH algorithm is that the spectrum excluding the common channels $(\mathcal{C} - \overline{\mathcal{C}})$ is divided between all the

users so that the users hop on the common channels as well as a few of the channels from the $(\mathcal{C} - \overline{\mathcal{C}})$. Therefore, there are different users hopping different channels in the same time-slot. When the new user joins the network, the rendezvous can be achieved faster as there are more channels being hopped at the same time, where rendezvous can posibilly occur.

When the $(k+1)^{th}$ user arrives on the k-user network, as the spectrum is divided equally, number of channels (denoted by N_c) hopped by the k users is equal *i.e.,* $N_i = N_j$ for $i, j \epsilon k$.

In phase (1), similar to the IIH, the probability for the $(k+1)^{th}$ user to achieve rendezvous with any of the k users is $p = \frac{k \cdot G_{k+1}}{N_c \cdot N_{k+1}}$. Unlike IIH, as the users in the DCH have working set containing more channels that $\overline{\mathcal{C}}$, in the best case, the k users can be hopping on all the N_{k+1} channels. Therefore, it is possible to achieve $p = \frac{G_{k+1}}{N_c}$ when $k = N_{k+1}$. Therefore the expected TTR = $\frac{1}{p} = \frac{N_c}{G_{k+1}}$ for phase 1, which is lower that the TTR for the IIH as N_c is formed after division of the channels and therefore $N_c \leq N_{k+1}$.

In phase (2), as the k users have their working set containing $\overline{\mathcal{C}}$, they can spread the information about the $(k+1)^{th}$ user. The completion time for the phase (2), will be longer than that for IIH because the working set for all k users also contain few more channels that are not common.

DCH algorithm based on above intuition and also has two components similar to IIH: 1) Working set modifications, 2) User count and information.

*1) **Working set modifications**:* $\overline{\mathcal{C}_i}$ denote the core channels for user L_i. If L_i has achieved rendezvous with other (but not all) k users, then the core channels for L_i are the channels common between L_i and other k users.

$\overline{\mathcal{C}_{ij}}$ is a new term that is used for the common channels between the users i and j excluding $\overline{\mathcal{C}_i}$. Note that, $\overline{\mathcal{C}_i}$ and $\overline{\mathcal{C}_{ij}}$ are different; $\overline{\mathcal{C}_{ij}}$ is a set of channels common only between user i and j excluding $\overline{\mathcal{C}}$. The working set is modified in the following way:

a) Core channels: When two users rendezvous pairwise, both users recalculate their core channels. The updated set of core channels for user i (and user j) becomes $\overline{\mathcal{C}_i} \cap \overline{\mathcal{C}_j}$. As a user rendezvous with more and more users, the set of core channels converge to $\overline{\mathcal{C}}$.

b) Mutually common channels: Whenever user i and j achieve rendezvous pairwise, they split the channels from the set $\overline{\mathcal{C}_{ij}}$ equally. If $\overline{\mathcal{C}_{ij}}$ has odd number of channels, then the user with lower number of channels get the one channel more than the other user. Splitting the mutually common channels make the two users to hop those channels separately, but collectively every channel is hopped.

c) Balancing the channels: The two users also balance the channels *i.e.,* if user L_i has more channels than user L_j, then user L_j can take some of the channels from L_i. L_j can take the channels that are available to L_j but are not being hopped by L_j, instead, they are being hopped by L_i.

*2) **User count and information**:* Similar to the IIH, when two users achieve pairwise rendezvous, they exchange the information about other users they have previously achieved rendezvous to accelerate the rendezvous among all the users.

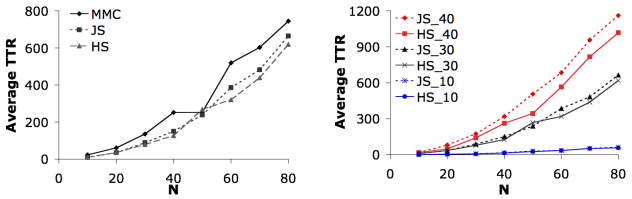
§ DIVIDE AND CONQUER HOPPING

1. **Input**: N
2. Find smallest prime number P, such that $P > N$
3. Calculate L_B based on P
4. Calculate L_A based on the working set
5. **while** (not terminated)
6. $\quad c = HS(N, P, L_B, L_A, t)$
7. $\quad$ **if** (successful rendezvous)
8. $\quad\quad$ Update the core channels ($\mathcal{C}_i = \mathcal{C}_i \bigcap \mathcal{C}_j$)
9. $\quad\quad$ Balance the channels
10. $\quad\quad$ Split the mutually common channels
11. $\quad\quad$ **if** (change in working set)
12. $\quad\quad\quad$ Recalculate L_A
13. $\quad\quad$ **end if**
14. $\quad\quad$ Update the user-count
15. $\quad$ **end if**
16. **end while**
17. c = smallest channel in $\overline{\mathcal{C}_i}$

Fig. 5. Pseudo-code for DCH.

When a user has received information about all other users, it terminates the algorithm and hops on the smallest channel in its core channel set $\overline{\mathcal{C}_i}$. Note that, when all the users have information about all other users, the working set might be different but $\overline{\mathcal{C}_i}$ will be equal to $\overline{\mathcal{C}}$.

*3) **Algorithm**:* The algorithm is formally showed in the figure 5. In this algorithm, the users hop the channels according to the HS algorithm as shown on line 6. Whenever a pairwise rendezvous happens, both users exchange the working sets and formulate the new working set by balancing the channels and also by splitting the common channels. As the working set for the users may have been modified after this stage, the users recalculate the L_A sequence, as shown on line 12. In line 14, the users exchange the information about the other users they have already achieved rendezvous.



(a) $\theta = 30\%$ (b) different θ

Fig. 6. Comparing different algorithms for 2-user cases.

VI. EVALUATION

We implemented all the algorithms in C. For both 2-user and the multi-user case, we compare our algorithm against the Jump-Stay algorithm from [8].

A. 2-User Case

*1) **Methodology**:* For the 2-user case, we generate the simulation cases by varying the maximum number of available channels N and the delay between the arrival of 2 users. N is varied between 0 to 80 in the steps of 10. The delay is varied between $[0, 3 \cdot N)$ because the round time for the JS algorithm is $3 \cdot P$. However as the P is not known beforehand, we vary the delay until $3 \cdot N$ as $N < P$. Without loss of generality,

we assume that user L_1 arrives in the 1^{st} time-slot, whereas user L_2 arrives after some delay.

For the 2-user case we observe the TTR for all values of delay and calculate the average. The TTR is measured from the time-slot when the last user joined the network.

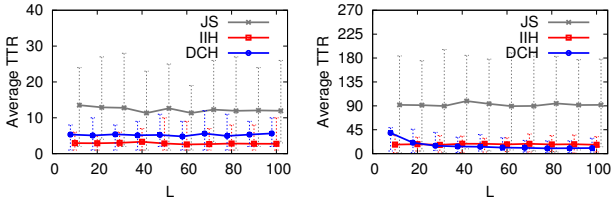
We consider asymmetrical model for evaluation as the solution for the symmetrical model is trivial.

*2) **Asymmetrical model**:* In the asymmetrical case, both users have different working sets. However, for successful rendezvous, they have one channel in common. In our evaluation, this common channel is randomly chosen. Also, we assume that both users have the same ratio of working set to the maximum available channels ($\frac{C_i}{C}$) denoted by θ ($0 < \theta < 1$).

Figure 6(a) shows the average TTR for different values of N for HS, JS and MMC [14] algorithms for $\theta = 0.3$.

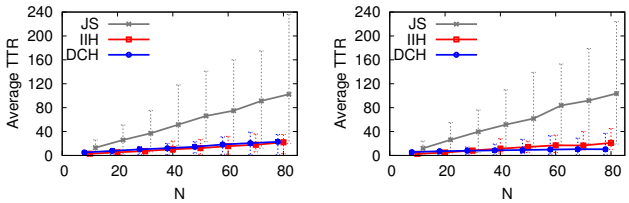
It can be seen that HS performs similar to the MMC and JS. However, the MMC algorithm does not provide any rendezvous guarantee, whereas JS and HS does. Therefore, we conclude that the performance of HS is better than MMC and similar to the JS in terms of guarantee.

Figure 6(b) compares the performance of the JS and HS algorithms for different θ. The performance is measured for $\theta = 0.1, 0.3$ and 0.4. The dotted curves correspond to the JS and the solid curves correspond to the HS. It can be seen that HS outperforms JS in most of the cases. The TTR for the HS is lowered by up to 40%, for $\theta = 0.4$.



(a) N = 10 (b) N = 70

Fig. 7. Comparing TTR for different algorithms, varying L. The TTR values for the same L are shown at an offset for clarity.



(a) L = 20 (b) L = 100

Fig. 8. Comparing TTR for different algorithms, varying the number of available channels per user. The TTR values for the same N are shown at an offset for clarity.

B. Multi-User Case

*1) **Methodology**:* For the multi-user case, we vary both the total number of available channels (N) and the total number of users (L). For every combination of L and N, we simulate 100 different cases by varying the arrival time of the users. The arrival time for the first user is 0 and the arrival time for other users is randomly chosen between $3 \cdot N$. The TTR is calculated from when the last user arrived on the network.

Similar to the asymmetrical model in the 2-user case, we assume that all the users have same $\theta = 0.3$.

For successful rendezvous, we assume that there is only one channel common across all the users. It should be noted that there can be multiple channels mutually common between any two users but there is only one channel that is common across all the users. The common channel and the working set for all the users are randomly chosen making sure that there is only one common channel across all the users.

a) Comparing IIH and DCH against JS: Now we compare the performance of the IIH, DCH and JS algorithms.

Figure 7(a) and 7(b) compare the average and observed maximum TTR for different values of L for $N = 10$ and $N = 70$ respectively.

It can be seen that for the fixed N, the average TTR does not increase with increasing L for all the three algorithms. However, TTR for the IIH and DCH is significantly lower than the TTR for the JS. In the average case, the TTR for the DCH is lowered by 77% compared to JS. The minimum and the maximum gain of DCH over JS are 52% and 90% respectively. The gain of DCH over JS increases as the L or N increase. Also, the maximum TTR for the IIH and DCH is significantly lower compared to JS.

Now, we evaluate the performance by fixing L and varying N. Figure 8(a) and 8(b) compare the average and maximum TTR for different values of N for $L = 20$ and $L = 100$.

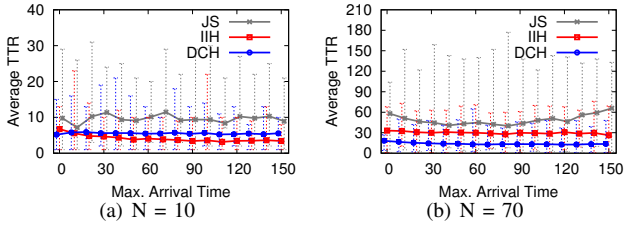
It can be seen that the average TTR for JS increases rapidly for increasing N. The TTR for the IIH also increases with respect to N but at a slower rate compared to JS. The TTR for the DCH increases with increasing N at the lower values of L but it remains constant at the higher values of L. This shows the scalability of our algorithms as N and L increase.

b) Evaluating the performance for different delay values: We then compare the performance of all the three algorithms with respect to changing the arrival time. In this scenario, we vary the maximum arrival time keeping the L and N fixed. The maximum arrival time varies between 0 to 150 time-slots in the increment of 10.

We first evaluate the TTR for different values of N keeping $L = 50$. Figure 9(a) and 9(b) compare the results for $N = 10$ and $N = 70$, respectively.

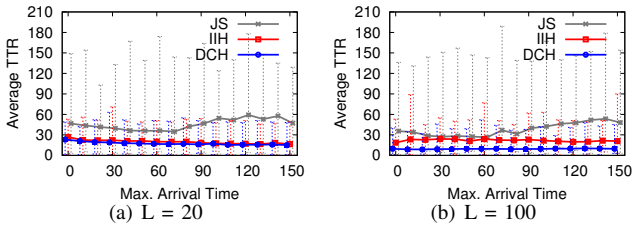
We then evaluate the TTR for different values of L keeping $N = 50$. Figure 10(a) and 10(b) compare the results for $L = 20$ and $L = 100$, respectively.

It can be seen that IIH and DCH are more resilient to the arrival time as the TTR is not affected significantly by the arrival time of the users, whereas the JS algorithm performs differently at different arrival time.



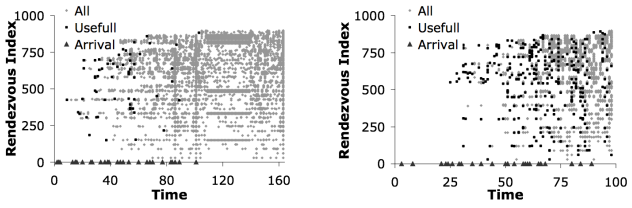
(a) N = 10 (b) N = 70

Fig. 9. Comparing TTR against arrival time for L = 50. The TTR values for the same max. arrival time are shown at an offset for clarity.



(a) L = 20 (b) L = 100

Fig. 10. Comparing TTR against arrival time for N = 50. The TTR values are for the same max. arrival times are shown at an offset for clarity.



(a) JS (b) DCH

Fig. 11. Rendezvous Points for JS and DCH.

c) Stability: We also evaluate the *stability* of the algorithms. Stability is evaluated using the TTR measured from the time-slot when the last (L^{th}) user arrived on the CRN when previous $L-1$ users have achieved the rendezvous.

To evaluate the stability, we find out more about the rendezvous pattern for the JS and DCH. Figure 11 shows the rendezvous pattern for JS and DCH when $L = 30$ and $N = 30$. In this case L^{th} user arrives in the next time-slot from when the previous $L-1$ users have achieved the rendezvous. The $L-1$ users arrive in the interval 0 to $3 \cdot N$.

The X-axis denotes the time and the Y-axis denotes the rendezvous index. The rendezvous index is calculated as $30 \cdot i + j$, when users L_i and L_j $(i > j)$ achieve rendezvous. As $L = 30$, rendezvous index is unique for every pair of users.

The gray dots are marked whenever a pair of users achieve rendezvous. If X users achieved rendezvous at the same channel in the same time-slot, then $\binom{X}{2}$ points are marked. The black dots are marked, whenever a *useful* information is gained during the rendezvous. In the JS, the useful information means if the at least one of the users change its 3-tuple (r_0, i_0, t) [8]. In the DCH algorithm, the useful information means, whenever there is a change in the working set or in case the user count increases. The arrival time for all the users is marked by gray triangles on the X-axis.

It can be seen that the DCH took only 9 time-slots, whereas JS required 62 time-slots to achieve rendezvous. For the DCH, the useful rendezvous points are scattered from beginning to the end. For the JS, the rendezvous points in the last 59 time-slots did not exchange any useful information. Therefore, it can be seen that DCH is more stable than JS.

VII. Conclusion

In this paper, we observe that the state of the art algorithm is not optimized for the multi-user scenarios. In particular, when two users encounter, one user inherits the other users hopping sequence but the sequence is never *shortened* or *split* among the encountering users. We introduce a new class of distributed algorithms for multi-user blind rendezvous, called Coordinated Channel Hopping (CCH), where users adjust or coordinate their sequence of channels being hopped upon pairwise rendezvous. In particular, Iterative Intersection Hopping takes the intersection of the two sequences, and Divide and Conquer Hopping splits the channels common to the two sequences, upon each pairwise rendezvous. Compared to existing rendezvous algorithms, our algorithms achieve 80% lower Time To Rendezvous (TTR) in case of multiple users.

References

[1] C. Arachchige, S. Venkatesan, and N. Mittal. An asynchronous neighbor discovery algorithm for cognitive radio networks. In *Proc. of IEEE DySPAN*, pages 1 –5, oct. 2008.

[2] M. M. Buddhikot, P. Kolodzy, S. Miller, K. Ryan, and J. Evans. Dimsumnet: New directions in wireless networking using coordinated dynamic spectrum access. In *in IEEE WoWMoM05*, pages 78–85, 2005.

[3] T. Chen, H. Zhang, M. Katz, and Z. Zhou. Swarm intelligence based dynamic control channel assignment in cogmesh. In *Communications Workshops, 2008. ICC Workshops '08. IEEE International Conference on*, pages 123 –128, may 2008.

[4] J. Jia, Q. Zhang, and X. Shen. Hc-mac: A hardware-constrained cognitive mac for efficient spectrum management. volume 26, pages 106 –117, jan. 2008.

[5] M.-R. Kim and S.-J. Yoo. Distributed coordination protocol for common control channel selection in multichannel ad-hoc cognitive radio networks. In *Proc. of IEEE WiMob*, 2009.

[6] Y. Kondareddy, P. Agrawal, and K. Sivalingam. Cognitive radio network setup without a common control channel. In *Proc. of IEEE MILCOM*, pages 1 –6, nov. 2008.

[7] L. Lazos, S. Liu, and M. Krunz. Spectrum opportunity-based control channel assignment in cognitive radio networks. In *Proc. of IEEE SECON*, pages 135–143, 2009.

[8] Z. Lin, H. Liu, X. Chu, and Y.-W. Leung. Jump-stay based channel-hopping algorithm with guaranteed rendezvous for cognitive radio networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 2444 –2452, april 2011.

[9] H. Liu, Z. Lin, X. Chu, and Y.-W. Leung. Ring-walk based channel-hopping algorithms with guaranteed rendezvous for cognitive radio networks. In *Green Computing and Communications (GreenCom)*, pages 755 –760, dec. 2010.

[10] L. Ma, X. Han, and C.-C. Shen. Dynamic open spectrum sharing mac protocol for wireless ad hoc networks. In *Proc. of IEEE DySPAN*, pages 203 –213, nov. 2005.

[11] J. Perez-Romero, O. Salient, R. Agusti, and L. Giupponi. A novel on-demand cognitive pilot channel enabling dynamic spectrum allocation. In *Proc. of IEEE DySPAN*, april 2007.

[12] S. Romaszko and P. Mähönen. Quorum-based channel allocation with asymmetric channel view in cognitive radio networks. In *Proceedings of the 6th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, pages 67–74, 2011.

[13] Y. Song and J. Xie. Prospect: A proactive spectrum handoff framework for cognitive radio ad hoc networks without common control channel. 2011.

[14] N. Theis, R. Thomas, and L. DaSilva. Rendezvous for cognitive radios. In *IEEE Transactions on Mobile Computing*, volume 10, pages 216 –227, feb. 2011.

[15] S.-U. Yoon and E. Ekici. Voluntary spectrum handoff: A novel approach to spectrum management in crns. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1 –5, may 2010.

[16] S.-U. Yoon, R. Murawski, E. Ekici, S. Park, and Z. Mir. Adaptive channel hopping for interference robust wireless sensor networks. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1 –5, may 2010.

[17] Y. Yuan, P. Bahl, R. Chandra, P. Chou, J. Ferrell, T. Moscibroda, S. Narlanka, and Y. Wu. Knows: Cognitive radio networks over white spaces. In *Proc. of IEEE DySPAN*, pages 416 –427, april 2007.

[18] J. Zhao, H. Zheng, and G.-H. Yang. Distributed coordination in dynamic spectrum allocation networks. In *Proc. of IEEE DySPAN*, pages 259 –268, nov. 2005.