Jump-Stay Rendezvous Algorithm for Cognitive Radio Networks

Hai Liu, Member, IEEE, Zhiyong Lin, Xiaowen Chu, Senior Member, IEEE, and Yiu-Wing Leung, Senior Member, IEEE

Abstract—Cognitive radio networks (CRNs) have emerged as an advanced and promising paradigm to exploit the existing wireless spectrum opportunistically. It is crucial for users in CRNs to search for neighbors via rendezvous process and thereby establish the communication links to exchange the information necessary for spectrum management and channel contention, etc. This paper focuses on the design of algorithms for blind rendezvous, i.e., rendezvous without using any centralized controller and common control channel (CCC). We propose a jump-stay channel-hopping (CH) algorithm for blind rendezvous. The basic idea is to generate CH sequence in rounds and each round consists of a jump-pattern and a stay-pattern. Users “jump” on available channels in the jump-pattern while “stay” on a specific channel in the stay-pattern. We prove that two users can achieve rendezvous in one of four possible pattern combinations: jump-stay, stay-jump, jump-jump, and stay-stay. Compared with the existing CH algorithms, our algorithm has the overall best performance in various scenarios and is applicable to rendezvous of multiuser and multihop scenarios. We derive upper bounds on the maximum time-to-rendezvous (TTR) and the expected TTR of our algorithm for both 2-user and multiuser scenarios (shown in Table 1). Extensive simulations are conducted to evaluate the performance of our algorithm.

Index Terms—Cognitive radio, blind rendezvous, channel hopping.

1 INTRODUCTION

The recent extensive measurements indicate that a majority of licensed spectrum is underutilized or unused in most of the time, which results in a significant amount of spectrum holes. In contrast, the unlicensed spectrum has already been overcrowded due to the exponential growth of various wireless devices. To solve the problem, dynamic spectrum access (DSA) techniques have been proposed with cognitive radio networks (CRNs) to alleviate the severe shortage of useable spectrum as well as improve the efficiency in the usage of licensed spectrum [1]. In the context of CRNs, the owner of a licensed channel is referred to as a primary user (PU) and other users of the channel are referred to as secondary users (SUs) or cognitive users (CUs). Each CU is equipped with one or more cognitive radios, which are capable of opportunistically identifying vacant portions of the spectrum (i.e., idle channels) and hopping between them without causing interference to the PUs of the spectrum. Unless otherwise specified, the users mentioned hereafter in this paper refer to CUs by default.

Users in a CRN should detect the presence of each other to establish communication links, so that information exchange, spectrum management, and data communication can be carried out. The process of two or more radios of users to meet and establish a link on a common channel is referred to as “rendezvous.” Rendezvous is a fundamental and essential operation in CRNs. However, implementation of rendezvous is nontrivial since users are even not aware of the presence of each other before rendezvous and available channels of each user are usually different and change dynamically.

Most of existing works on rendezvous either utilized centralized controller [2], [3] or employed dedicated common control channel (CCC) [4], [5], [6] to facilitate the rendezvous. Although these strategies simplify the rendezvous considerably, they suffer from new problems which include the low scalability and flexibility of centralized systems, the difficulty or even infeasibility in establishing a CCC, the vulnerability of CCC to jamming attack, and so on. Therefore, blind rendezvous systems without any centralized controller and dedicated CCC are preferable in practice. Channel-hopping (CH) is one of the most representative techniques for blind rendezvous. With CH technique, each user of a CRN selects a set of available channels and hops among these channels for rendezvous with its potential neighbors. If all users have the same available channels, we call it symmetric model. We call it asymmetric model otherwise, i.e., different users might have different available channels.

A number of CH algorithms [7], [8], [9], [10], [11], [12], [13] have been proposed in the recent literature. The most notable works are [12] and [11]. In [12], Theis et al. proposed modular clock (MC) and modified modular clock (MMC) for rendezvous of two users under the symmetric model and the asymmetric model, respectively. MC and MMC give very good performance in experimental simulations. Theoretically, however, both MC and MMC cannot guarantee the
rendezvous of two users in finite time and cannot be directly applied to rendezvous of multiple users. Shin et al. proposed a novel CH algorithm, called channel rendezvous sequence (CRSEQ) [11], which can guarantee rendezvous of two users under the asymmetric model and can be easily extended to be applied to rendezvous of multiple users. Note that any CH algorithm to the asymmetric model is also applicable to the symmetric model. Although CRSEQ performs well under the asymmetric model, we show later via both analysis and experiments that CRSEQ is not so efficient when it is applied to the symmetric model. Before rendezvous and exchange of messages with neighbors, users in CRN are usually not aware of the model which is actually suitable to the network. Without information of the symmetric/asymmetric model, users cannot choose appropriate rendezvous algorithms beforehand. Therefore, it is crucial for rendezvous algorithms to be self-adaptive and perform well in both models. Table 1 highlights the performance of our algorithm and other representative algorithms.

In this study, we focus on the design of CH algorithm for guaranteed rendezvous of two or multiple users in time-slotted CRNs. We aim to develop the rendezvous algorithm which is self-adaptive and performs well under both symmetric and asymmetric models. Contributions of this work are summarized as follows: 1) New algorithm with good performance under both the symmetric and asymmetric models: a new CH algorithm called jump-stay (JS) is proposed, which can achieve guaranteed rendezvous without the need of time synchronization, and performs well under both symmetric and asymmetric models. 2) Generalized rendezvous for multiuser and multihop scenarios: JS is applicable to rendezvous of multiple users within 1-hop and even multihop neighborhood. 3) Solid theoretical analysis: we derive an upper bound of maximum time-to-rendezvous (TTR) and expected TTR (E(TTR)) of JS for both 2-user and multiuser multihop scenarios, where TTR is defined as the number of time slots that it takes for users to achieve rendezvous.

The rest of this paper is organized as follows: Related work is reviewed in Section 2. The rendezvous problem is formally formulated in Section 3. We propose the JS algorithm and present its theoretical analysis in Section 4. Numerical simulation is conducted in Section 5. We conclude our work in Section 6.

2 RELATED WORK

Basically, the existing rendezvous systems in CRNs can be divided into two categories: centralized and decentralized.
In a centralized system, a centralized controller (also known as server or base station) is preset to assist the rendezvous of all other users in the network. In a decentralized system, users have to achieve rendezvous without aid of any centralized controller. Both centralized and decentralized rendezvous systems can be further classified into two subcategories depending on using CCC or not. In the following, we review the representative rendezvous systems based on this taxonomy.

- **Centralized systems.** Most of the centralized rendezvous systems, such as DIMSUMNet [2] and DSAP [3], require the server to operate over a preselected CCC, which is well known and accessible to all users in the network. Technically, the desired CCC can be simply allocated in unlicensed bands or even in licensed bands. Using CCC makes convenient the communication between the server and other users and thus simplifies the rendezvous process significantly. However, using CCC may suffer from problems which include:
  - allocating CCC in unlicensed bands will just aggravate their congestions;
  - allocating and maintaining CCC in licensed bands is hard or even impracticable due to the changing and diverse channel availability of each user;
  - CCC is vulnerable to jamming attack and easily acts as a single point of failure.

In [14], an exhaustive search-based protocol and a random protocol were proposed for rendezvous of users without using CCC.

- **Decentralized systems using CCC.** Though the centralized systems are relatively simple to implement, its scalability and robustness is poor since the server tends to become the bottleneck of communications when the size of the network increases. Among the decentralized systems using CCC, works in [5] and [6] assume a global CCC which is obtained in advance and is accessible to all users. Obtaining global CCC in DSA environment is difficult or even infeasible due to diversity and dynamic of available channels of each user. Therefore, seeking local rather than global CCC would be more feasible in CRNs. Works in [15] and [16] focus on establishing the local CCC-based systems, where users are clustered according to their available channel sets such that the users of the same cluster can access a local CCC. To form a cluster, however, the neighboring users are required to coordinate mutually which essentially relies on the implementation of rendezvous. Moreover, the overhead of establishing and maintaining a local CCC is considerable [17].

- **Decentralized systems without using CCC (blind rendezvous).** Since using CCC has its own limitations, decentralized rendezvous systems without using CCC, i.e., blind rendezvous systems, draw more attentions of researchers. A typical approach for blind rendezvous is CH technique in which each user selects a set of available channels and hops among these channels for rendezvous with potential neighbors. The rendezvous is achieved if the users hop on a commonly available channel in the same time slot. We focus on the works falling into this category in the following:

  - **Random algorithms.** A trivial CH algorithm is to let each user decide its own hopping sequence based on the available channels in a purely random way. An improved random algorithm, called AMRCC, was proposed in [9]. The basic idea of AMRCC is that the channels with lower interference to PUs have larger chances to be selected into the CH sequence. However, the random algorithms including AMRCC cannot guarantee the rendezvous of users in finite time.

  - **Algorithms that require time synchronization.** If time synchronization is achievable in the system, there are several CH algorithms which achieve guaranteed rendezvous. Bahl et al. presented a pioneering work in [7] and proposed a link-layer protocol, named SSCH. With SSCH, each user is allowed to select multiple (channel, seed) pairs and the CH sequence is determined based on these pairs. Though SSCH is designed for increasing the capacity of IEEE 802.11 networks, it guarantees rendezvous of users under the symmetric model. Authors in [18] proposed a deterministic approach in which each user is scheduled to broadcast on every channel in an exhaustive manner. Based on quorum systems, Bian et al. proposed in [8] two CH algorithms, namely M-QCH and L-QCH, which can guarantee rendezvous between users in time-synchronized systems. Another CH algorithm called A-QCH was proposed in [8] for the asynchronous systems. However, A-QCH is only applicable to systems with two channels, which limits the applicability of the algorithm.

  - **Algorithms that work without time synchronization.** There are CH algorithms which do not require time synchronization. A representative one with guaranteed rendezvous for asynchronous systems was proposed in [10] and was later referred to as the generated orthogonal sequence (GOS) algorithm in [12]. GOS is only applicable to the symmetric model. In our previous work [19], we presented a ring-walk (RW) algorithm which guarantees the rendezvous under both models. In RW, each channel is represented as a vertex in a ring. Users walk on the ring by visiting the vertices (channels) with different velocities and rendezvous is guaranteed since users with lower velocities will be caught by users with higher velocities. However, RW requires that each user has a unique ID and knows the upper bound of network size. Recently, a notable work by Theis et al. [12] presented two CH algorithms: modular clock algorithm (MC) and its modified version MMC for symmetric model and asymmetric model, respectively. The basic idea of MC and MMC is that each user picks a proper prime number and randomly selects a rate less than the prime number. Based on the two parameters, the
The PUs are the owners of some licensed spectrum, which coexist with one or more PUs in the same geographical area. We consider a CRN consisting of

### SYSTEM MODEL AND PROBLEM FORMULATION

The rendezvous processes such as beaconing on a prescheduled channel, sensing can be found in [22]. Let $G$ denote the number of common available channels of all users, i.e., $G = \bigcap_{i=1}^{K} C_i$. We consider the following two models:

1. **Symmetric model.** All users have the same available channels if their geographical locations are close. That is, for any $1 \leq i, j \leq K$, we have $C_i = C_j$. For simplicity, we assume $C_i = C_j = C$.

2. **Asymmetric model.** Different users might have different available channels if their geographical locations are far. We assume that at least one common channel is shared by all users in the network, i.e., $G \neq 0$. This is a necessary assumption since there is no feasible solution, otherwise, for rendezvous of the users.

As in most of the existing works on CH algorithms [11], [12], we assume that the network is time slotted and all time slots are with the same and fixed length. In each time slot, each user hops on a channel to attempt rendezvous with its potential neighbors. Our goal is to seek the CH algorithm which is applicable to both the symmetric and asymmetric models without the need of time synchronization (i.e., each user may start its channel hopping at any time). In practice, a successful rendezvous (i.e., exchanging information between users successfully) involves many detailed processes such as beaconing on a prescheduled channel, applying a specific handshaking mechanism, etc. In this work, we simplify the rendezvous problem and focus on the design of CH algorithms. More specifically, we assume

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Need time-synchronization</th>
<th>Need additional information</th>
<th>Work under asymmetric model</th>
<th>Users use the identical sequence</th>
<th>Guaranteed rendezvous under symmetric model</th>
<th>Guaranteed rendezvous under asymmetric model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our algorithm</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Pure random [12]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ARMMC [9]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[18]</td>
<td>Yes</td>
<td>User ID and network size</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>M-/L-QCH [8]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SSCH [7]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GOS [10]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MC [12]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MMC [12]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DRSEQ [13]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CRSEQ [11]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RW [19]</td>
<td>No</td>
<td>User ID and network size</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

user generates its CH sequence via predefined modulo operations. Although MC and MMC are shown to be effective, both algorithms cannot guarantee the rendezvous if the selected rates or the prime numbers of two users are identical. Yang et al. proposed two significant algorithms, namely deterministic rendezvous sequence (DRSEQ) [13] and channel rendezvous sequence [11], which provide guaranteed rendezvous for the symmetric model and the asymmetric model, respectively. Given the number of channels $M$, a DRSEQ exactly consists of $(2M + 1)$ indices, which can be expressed as “$1, 2, \ldots, M, e, M, M - 1, \ldots, 1$,” where $e$ denotes a NULL item. In CRSEQ, the sequence is constructed based on triangle numbers and modulo operations. In terms of maximum TTR (MTTR), CRSEQ is quite good under the asymmetric model but it does not perform well under the symmetric model. Our experiment results in Section 5 show that CRSEQ does not perform well in terms of the average TTR regardless of its small MTTR.

Though very few, there are still some other blind-rendezvous systems not based on the CH techniques. For instance, works in [20] and [14] tried to set up an infrastructure by electing a leader in the decentralized network, which is responsible for discovering its neighboring users and assisting their rendezvous. In [21], some special signals such as cyclostationary signatures are employed to facilitate the rendezvous of users.

Table 2 summarizes the characteristics of existing representative CH algorithms.

### Summary of Representative CH Algorithms

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Need time-synchronization</th>
<th>Need additional information</th>
<th>Work under asymmetric model</th>
<th>Users use the identical sequence</th>
<th>Guaranteed rendezvous under symmetric model</th>
<th>Guaranteed rendezvous under asymmetric model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our algorithm</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Pure random [12]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ARMMC [9]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[18]</td>
<td>Yes</td>
<td>User ID and network size</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>M-/L-QCH [8]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SSCH [7]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>GOS [10]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>MC [12]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>MMC [12]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DRSEQ [13]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CRSEQ [11]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RW [19]</td>
<td>No</td>
<td>User ID and network size</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
that the rendezvous of $K$, $K \geq 2$, users is said to be successful if all the users hop on a commonly available channel in the same time slot. Since time synchronization is not available in the network, “the same time slot” of two CH sequences means that the overlap of two time slots is sufficient to complete all necessary steps for rendezvous. In this sense, the CH sequences are equivalent to be slot-aligned even without the time synchronization. Take example in Fig. 1a, the overlap of two CH sequences, marked in gray, is long enough to complete the rendezvous process. Thus, the two CH sequences are equivalent to be slot-aligned, shown in Fig. 1b. Let $T$ be the duration for exchanging the necessary control messages in rendezvous. Since the cognitive users may not be time synchronized, each time slot should have a duration of $2T$ [13]. According to IEEE 802.22 (which is a standard for using cognitive radio), $T$ is equal to 10 ms [23] and hence each time slot has a duration of 20 ms. Based on these assumptions, we formulate the rendezvous problem as follows:

**Rendezvous problem.** Given a multihop CRN consisting of $K$ ($K \geq 2$) users and $C$, which denotes the set of available channels of user $i$ ($i = 1, 2, \ldots, K$), to design CH algorithms to generate the CH sequences for the users, such that all users are guaranteed to hop on a commonly available channel in the same time slot, regardless of different time when the users start their CH sequences.

To explain the need of rendezvous, we consider an application example in which two or more users want to send files to each other using cognitive radio. There may be different users may find different channel availability because they are located at different positions relative to the PUs. These users must: 1) select a channel that is commonly available to them and 2) inform each other that they will use this channel for communication. After completing these operations, the users can send files to each other through the selected channel.

TTR is an important metric to evaluate CH algorithms. It is defined as the number of time slots that it takes for users to achieve rendezvous once all users have begun their hopping. The TTR of a CH algorithm is usually not constant due to random starting time of CH sequences of the users. Thus, maximum TTR and expected TTR are usually adopted to evaluate the performance of CH algorithms. If a CH algorithm has a finite MTTR, then it is said to be with guaranteed rendezvous. In this study, we seek CH algorithms with guaranteed rendezvous for both the symmetric and asymmetric models.

### 4 Jump-Stay Rendezvous Algorithm

In this section, we present the jump-stay rendezvous algorithms. We first study the algorithm for the rendezvous of two users in Section 4.1 and then extend our algorithm to the multiuser and multihop scenario in Section 4.2.

#### 4.1 Algorithm for 2-User Scenario

**4.1.1 Basic Idea**

Our jump-stay algorithm generates CH sequence in rounds, and each round consists of one *jump-pattern* and one *stay-pattern*. Jump-pattern and stay-pattern are specific segments of CH sequences. Intuitively, users continuously “jump” on available channels during the jump-pattern while “stay” on a specific channel during the stay-pattern. The user performs the jump-pattern first and the stay-pattern follows the jump-pattern in each round.

To generate the two patterns in a round, the user is required to select three parameters beforehand: $P$ which is the smallest prime number greater than $M$, a nonzero number $r$ from $[1, M]$ and an index $i$ from $[1, P]$. In each round, the jump-pattern lasts for $2P$ time slots and the subsequent stay-pattern lasts for $P$ time slots (That is, each round takes $3P$ time slots in total). In the jump-pattern, the user starts with index $i$ and keeps jumping (hopping) in $[1, P]$ with step-length $r$ by using the modulo operations on $P$. In the subsequent stay-pattern, the user just stays on channel $r$ which is the step-length of this round.

The following *JSHopping* function describes how to determine the hopping channel at time slot $t$.

```
Function JSHopping
1: Input: $M$, $P$, $r$, $i$, $t$
2: Output: channel $c$
3: $t = t \mod 3P$; //each round takes 3P timeslots
4: if ($t < 2P$) $j = ((i + tr - 1) \mod P) + 1$; /jump pattern
5: else $j = r$; //stay pattern
6: end
7: if ($j > M$) $j = (j - 1) \mod M + 1$; end //remapping
8: return $c = c_j$
```

In line 1, variable $t$ is a time slot counter starting from 0. In line 3, “mod” is the modulo operation. In line 7, there is a remapping operation if the generated channel index $j$ exceeds $M$ (This is possible since $P$ is greater than $M$).

An illustration example. Suppose $M = 4$, $r = 1$, and $i = 3$ in the current round. $P$ is equal to 5 which is the smallest prime number greater than 4. The complete hopping sequence generated by *JSHopping* in this round is as follows (the numbers in parentheses are the channel indices before the remapping operation):

$3, 4, 1(5), 1, 2, 3, 4, 1(5), 1, 2, 1, 1, 1, 1, 1$.

Note that in the above sequence, the first 10 numbers in italic are corresponding to the jump-pattern while the remaining five numbers are corresponding to the stay-pattern. In the jump-pattern, the user starts with $i = 3$ and jumps in $[1, 2, 3, 4, 5]$ with step-length $r = 1$ under the modulo operation with respect to $P = 5$. That is, if the current index is $j$, then the next index equals $((j + r - 1) \mod P) + 1$ which may take only value from 1 to $P$. 

![Image showing the overlap of two CH sequences.(a) Slot-aligned CH sequences.](image-url)
4.1.2 Algorithm Description

In each round of 3P time slots, each user continuously calls JS\text{Hopping} for generating its hopping sequence until rendezvous is achieved. Among the three parameters in JS\text{Hopping}, i.e., $P$, $r$, and $i$, the prime number $P$ always remains the same in all rounds, while step-length $r$ and starting-index $i$ should be adjusted properly according to the following rules so as to guarantee the rendezvous of users:

1. Step-length $r$ takes an integer in $[1, M]$.
2. In each round of 3P time slots, $r$ remains the same. In the next round, $r$ is switched to next number in $[1, M]$ in round-robin fashion.
3. Starting-index $i$ takes an integer in $[1, P]$.
4. In consecutive $M$ rounds of 3MP time slots, $i$ remains the same. $i$ is switched to next number in $[1, P]$ every $M$ rounds in round-robin fashion.

The CH algorithm for rendezvous of two users is formally presented as follows:

\begin{algorithm}
\caption{Algorithm JS\_2}
\begin{algorithmic}
\STATE 1: \textbf{Input}: $M$, $C_k$ //for the k-th user
\STATE 2: $P=$ the smallest prime number greater than $M$;
\STATE 3: $r_0$=\text{rand}[1, M]; $i_0$=\text{rand}[1, P]; $t$=0;
\WHILE{not rendezvous}
\STATE 5: $n$=\text{rand}(1, 3P); $r$=((i_0n-1) mod M)+1;
\STATE 6: $m$=\text{rand}(1, 3MP); $i$=((i_0n-1) mod P)+1;
\STATE 7: $c$=JS\text{Hopping}($M$, $P$, $r$, $i$, $t$);
\STATE 8: if $c \notin C_k$
\STATE 9: $C_k$=RandomSelect($C_k$);
\STATE 10: end
\STATE 11: $t$=$t$+1;
\STATE 12: Attempt rendezvous on channel $c$;
\STATE 13: end
\end{algorithmic}
\end{algorithm}

In line 3, function “\text{rand}” is to randomly generate an integer in the specified interval according to a uniform distribution. The initial value of step-length $r_0$ and starting-index $i_0$ are set by employing the rand function. In line 5, step-length $r$ is updated every round (i.e., every 3P time slots). As a step-length for hopping, $r$ should be kept nonzero. In any consecutive $M$ rounds, $r$ will visit all the integers in $[1, M]$. In line 6, starting-index $i$ is updated every $M$ rounds (i.e., every 3MP time slots). Compared with step-length, starting-index is less often updated. In lines 8-10, if the generated channel by JS\text{Hopping} is not available, the user replaces it with an available channel which is randomly selected from its available channel set (function RandomSelect in line 9). This operation is called random-replace which is to fully utilize the time slots and increase the chance for rendezvous. Similar random-replace operation could be found in MMC [12].

The CH sequence generated by JS\_2 is illustrated in Fig. 2. Note that step-length $r$ is updated every round of 3P time slots while starting-index $i$ is updated every $M$ rounds of 3MP time slots. The whole sequence is in a two-layer structure, where the inner-round is the basic round of 3P time slots and the outer-round consists of exact $M$ inner-rounds of 3MP time slots. Step-length is updated every inner-round while starting-index is updated every outer-round.

Fig. 3 shows rendezvous of two users by using JS\_2. Suppose that $|C|=M=2$ and $P=3$. Each inner-round consists of nine time slots and each outer-round consists of 18 time slots. Users 1 and 2 generate their CH sequences with ($i_0=2$, $r_0=1$) and ($i_0=1$, $r_0=2$), respectively. Figs. 3a and 3b show CH sequences of users 1 and 2, respectively. For ease of following the CH sequences, we present both sequences with and without remapping operation. Fig. 3c shows rendezvous of the two users without time synchronization in several scenarios.

4.1.3 Algorithm Analysis

In this section, we present rigorous theoretical analysis of our JS\_2 algorithm. Specifically, we derive the upper bounds of MTTR of JS\_2 in Theorems 1 and 2 under the symmetric model and the asymmetric model, respectively, and thus prove its correctness. We prove that JS\_2 can provide guaranteed rendezvous in one of three possible pattern combinations: \text{jump-stay} (stay-jump is equivalent to jump-stay), \text{jump-jump} and \text{stay-stay}, where the jump-stay type means that rendezvous is achieved in the overlap between jump-pattern of one user with stay-pattern of the other user (jump-jump type and stay-stay type could be defined in similar way). Based on the results, we further present the upper bounds of E(TTR) of JS\_2 under both the symmetric and asymmetric models.

To analyze the MTTR of JS\_2 under the symmetric model, we have the following lemmas.

\textbf{Lemma 1.} Given a positive integer $P$, if integer $r \in [1, P)$ is relatively prime to $P$ (i.e., the common factor between them is 1), then for any integer $x \in [0, P)$ the sequence

\[ S = <(x \mod P) + 1, ((x + r) \mod P) + 1, \ldots, ((x + (P - 1)r) \mod P) + 1> \]

is a permutation of $<1, 2, \ldots, P>$. 

Fig. 2. The CH sequence of JS\_2.

Fig. 3. Rendezvous of two users by using JS\_2.
The Proof of Lemma 1 can be found in Section I in the supplemental material available online.

Lemma 1 implies that all the available channels are visited in any consecutive $P$ time slots in the jump-pattern (line 4 of the JShopping function).

**Lemma 2.** Given a prime number $P$, if $r_1$ and $r_2$ are two different integers in $[1, P]$, then for any integers $x_1 \in [0, P)$ and $x_2 \in [0, P)$, there must exist an integer $k \in [0, P)$ such that $(x_1 + kr_1) \mod P = (x_2 + kr_2) \mod P$.

The Proof of Lemma 2 can be found in Section I in the supplemental material available online.

Lemma 2 implies that rendezvous of two users is guaranteed if both users are currently in the jump-pattern with different step-lengths ($r_1$ and $r_2$ in Lemma 2) and the overlap between their jump-patterns is not less than $P$ time slots (Lemma 2). An illustration example for Lemma 2 can be found in the JShopping function (presented later in this section).

Based on Lemmas 1 and 2, we prove correctness of the JS_2 algorithm and derive the upper bound of MTTR under the symmetric model in the following theorem.

**Theorem 1.** Under the symmetric model, any two users performing JS_2 achieve rendezvous in at most $3P$ time slots, where $P$ is the smallest prime number greater than $M$.

**Proof.** Without loss of generality, we assume that user 1 begins performing JS_2 not later than user 2. When user 2 starts the hopping, it should be in jump-pattern but user 1 may be in jump-pattern or stay-pattern. Thus, we discuss the following two cases.

Case 1. User 1 is in jump-pattern. Let $l$ denote the overlap between the jump-patterns of both users. This case can be further divided into three subcases, which are illustrated in Figs. 4a, 4b, and 4c, respectively. The unshadowed blocks represent the jump-patterns and the shadowed blocks represent the stay-patterns. $r_i, i = 1, 2$, is the step-length currently used by user $i$ in its jump-pattern (unshadowed block) and is also equal to the channel index used in the subsequent stay-pattern (shadowed block).

Subcase 1a. $l \geq P, r_1 \neq r_2$ (Fig. 4a). Since the overlap between the jump-patterns of both users (i.e., $l$ in the figure) includes no less than $P$ time slots and $r_1 \neq r_2$, according to Lemma 2 we know that a rendezvous should occur during the first $P$ time slots of the overlap. That is, $TTR \leq P$.

Subcase 1b. $l \geq P, r_1 = r_2$ (Fig. 4b). In this subcase, $r_1 = r_2$ means that both users will stay on the same channel when they enter the stay-patterns. Therefore, rendezvous is achieved once user 2 enters the stay-pattern. That is, $TTR \leq 2P + 1$.

Subcase 1c. $l < P$ (Fig. 4c). In this subcase, $l < P$ implies that the overlap between the jump-pattern of user 2 and the stay-pattern of user 1 (i.e., $l'$ in the figure) contains at least $P$ time slots. According to Lemma 1, there is a permutation of $<1, 2, \ldots, P>$ in user 2’s sequence while user 1 stays on channel $r_1$ in $l'$. Thus, the rendezvous is achieved and $TTR \leq 2P$.

Case 2. User 1 is in stay-pattern. Similar to Case 1, this case could be divided into three subcases, which are illustrated in Figs. 4d, 4e, and 4f, respectively. In this case, $l$ denotes the overlap between the stay-pattern of user 1 and the jump-pattern of user 2. Since the length of the stay pattern is exact $P$ time slots, there is no case of $l > P$.

Subcase 2a. $l = P$ (Fig. 4d). According to Lemma 1, we have $TTR \leq P$.

Subcase 2b. $l > P, r_1 \neq r_2$ (Fig. 4e). Since the overlap $l'$ between the jump-patterns of two users exceeds $P$ time slots and $r_1 \neq r_2$, we have $TTR \leq 2P$ according to Lemma 2.
Subcase 2.c. \( l < P, r_1 = r_2 \) (Fig. 4f). Similar to Subcase 1.b, we have \( TTR \leq 3P \).

According to analysis of Cases 1 and 2, we prove that TTR of JS_2 is upper bounded by \( 3P \) time slots under the symmetric model.

Based on Theorem 1, we obtain the following corollary which provides an upper bound of \( E(TTR) \) of JS_2 under the symmetric model.

**Corollary 1.** Under the symmetric model, \( E(TTR) \) of JS_2 is not greater than \( 5P/3 + 3 \), where \( P \) is the smallest prime number greater than \( M \).

The Proof of Corollary 1 can be found in Section I in the supplemental material available online.

We have proved correctness of JS_2 and derived the upper bound of its MTTR under the symmetric model. Examples in Fig. 5 illustrate how JS_2 can provide guaranteed rendezvous of two users in one of jump-stay, jump-jump, and stay-stay types. Suppose that \( M = 2 \) and \( P = 3 \). Each inner-round consists of nine time slots and each outer-round consists of 18 time slots. To illustrate all possible cases of rendezvous, we introduce three users and each pair of the users demonstrates a type of rendezvous between two users. Users 1, 2, and 3 generate their CH and thus the rendezvous is actually achieved under the asymmetric model.

**Theorem 2.** Under the asymmetric model, any two users performing JS_2 achieve rendezvous in at most \( 3MP(P - G) + 3P \) time slots, where \( P \) is the smallest prime number greater than \( M \) and \( G \) is the number of commonly available channels between the two users.

The Proof of Theorem 2 can be found in Section I in the supplemental material available online.

The following corollary presents an upper bound of \( E(TTR) \) of JS_2 under the asymmetric model.

**Corollary 2.** Under the asymmetric model, \( E(TTR) \) of JS_2 is not greater than \( 2MP(P - G) + (M + 5 - P - (2G - 1)/M)P \), where \( P \) is the smallest prime number which is greater than \( M \) and \( G \) is the number of commonly available channels between the two users.

The Proof of Corollary 2 can be found in Section I in the supplemental material available online.

Until now, we are mainly concerned with theoretical performance of the proposed algorithm. In practice, however, the performance of our algorithm may be affected by realistic factors, which mainly include accuracy of spectrum sensing and dynamics of PU activities. In the following, we discuss the impact of these two factors on the performance of our algorithm:

- **Accuracy of spectrum sensing.** Each CU obtains the knowledge of available channels via sensing. While the spectrum sensing algorithms reviewed in the literature [22] can give good accuracy, they are not perfectly accurate and they may result in two undesirable consequences: i) **False alarm.** A channel is available but it is sensed to be unavailable, and ii) **Misdetection.** A channel is unavailable but it is sensed to be available. The existing sensing algorithms give small probability of false alarm or misdetection (when a PU is transmitting, the signal strength is significantly larger). When false alarm or misdetection happens, the proposed and existing rendezvous algorithms are affected in the same way as follows:

**False alarm.** When a channel is available but it is sensed to be unavailable, the CU has fewer choices and hence it may take longer time to achieve rendezvous. In the worst case that the CUs cannot
correctly identify a commonly available channel, they cannot achieve rendezvous.

Misdetection. When a channel is unavailable but it is sensed to be available, the CU would attempt to use this channel for rendezvous and hence it may take longer time to achieve rendezvous. In addition, when the CU attempts to use this channel, this produces interference to the PU.

• Dynamics of PU activities. The dynamics of the primary users can be taken into account using one of the following methods. Suppose the proposed algorithm attempts rendezvous on channel $c_1$ and then channel $c_2$.

- Sensing and rendezvous are performed in a sequential manner [12]. At the beginning of each time slot, the user senses the availability of a channel. If this channel is available, the user will attempt rendezvous on this channel in the remaining of the time slot [12].

- Sensing and rendezvous are performed in a pipelined manner. The user’s device has a sensing unit and a rendezvous unit. When the rendezvous unit attempts rendezvous on channel $c_1$, the sensing unit senses the availability of channel $c_2$. This method can speed up the rendezvous process compared with the first method.

In practice, the rendezvous process of our algorithm is not long and thus the effect of dynamics of PU activities is limited. For example, considering the scenario with 10 channels, 40 CUs and average network diameter $D$ of 12.02 (the resulting average TTR is the largest as shown in Fig. 12a in Section 5), we find that our algorithm can complete the rendezvous process in an average of 92.81 time slots or 1.86 s (a time slot lasts for 20 ms).

4.2 Algorithm for Multiuser Multihop Scenario

The JS_2 algorithm can be smoothly extended to the multiuser and multihop scenario. Our basic idea is to apply the pair-wise rendezvous of two users to achieve the global rendezvous of all users. Note that two users can exchange information over a commonly available channel if the rendezvous is successfully achieved on this channel. Once two users achieve rendezvous successfully, parameter setting of their algorithms can be synchronized, so that the two users will generate the identical CH sequence afterward. Recall that each user performing JS_2 keeps three parameters $r_0$, $i_0$, and $t$ which determine its CH sequence. Parameters $M$ and $P$ are identical to all users in the network. We use a 3-tuple $(r_0, i_0, t)$ to represent the three parameters and define the partial order of the 3-tuples. $(r_{i_0}, i_{q_0}, t_{i_0})$ is said to be smaller than $(r_{q_0}, i_{q_0}, t_{q_0})$ if and only if one of the following conditions is met: 1) $r_{i_0} < r_{q_0}$; 2) $r_{i_0} = r_{q_0}$ and $i_{i_0} < i_{q_0}$; 3) $r_{i_0} = r_{q_0}$, $i_{i_0} = i_{q_0}$, and $t_{i_0} < t_{q_0}$.

Our algorithm for rendezvous in multiuser and multihop scenario works as follows: Each user performs pairwise rendezvous with one of its potential neighbors. Once rendezvous of two users is successful, the two users exchange information of their 3-tuples. The user who holds the smaller 3-tuple will follow its neighbor by updating its 3-tuple with the larger one. Then, the two users use the identical and synchronized CH sequence for rendezvous with other users in similar way. This process is continued until rendezvous of all users is achieved (all users eventually use the same and the largest 3-tuple).

- Sensing and rendezvous are performed in a sequential manner [12]. At the beginning of each time slot, the user senses the availability of a channel. If this channel is available, the user will attempt rendezvous on this channel in the remaining of the time slot [12].

- Sensing and rendezvous are performed in a pipelined manner. The user’s device has a sensing unit and a rendezvous unit. When the rendezvous unit attempts rendezvous on channel $c_1$, the sensing unit senses the availability of channel $c_2$. This method can speed up the rendezvous process compared with the first method.

In practice, the rendezvous process of our algorithm is not long and thus the effect of dynamics of PU activities is limited. For example, considering the scenario with 10 channels, 40 CUs and average network diameter $D$ of 12.02 (the resulting average TTR is the largest as shown in Fig. 12a in Section 5), we find that our algorithm can complete the rendezvous process in an average of 92.81 time slots or 1.86 s (a time slot lasts for 20 ms).

4.2 Algorithm for Multiuser Multihop Scenario

The JS_2 algorithm can be smoothly extended to the multiuser and multihop scenario. Our basic idea is to apply the pair-wise rendezvous of two users to achieve the global rendezvous of all users. Note that two users can exchange information over a commonly available channel if the rendezvous is successfully achieved on this channel. Once two users achieve rendezvous successfully, parameter setting of their algorithms can be synchronized, so that the two users will generate the identical CH sequence afterward. Recall that each user performing JS_2 keeps three parameters $r_0$, $i_0$, and $t$ which determine its CH sequence. Parameters $M$ and $P$ are identical to all users in the network. We use a 3-tuple $(r_0, i_0, t)$ to represent the three parameters and define the partial order of the 3-tuples. $(r_{i_0}, i_{q_0}, t_{i_0})$ is said to be smaller than $(r_{q_0}, i_{q_0}, t_{q_0})$ if and only if one of the following conditions is met: 1) $r_{i_0} < r_{q_0}$; 2) $r_{i_0} = r_{q_0}$ and $i_{i_0} < i_{q_0}$; 3) $r_{i_0} = r_{q_0}$, $i_{i_0} = i_{q_0}$, and $t_{i_0} < t_{q_0}$.

Our algorithm for rendezvous in multiuser and multihop scenario works as follows: Each user performs pairwise rendezvous with one of its potential neighbors. Once rendezvous of two users is successful, the two users exchange information of their 3-tuples. The user who holds the smaller 3-tuple will follow its neighbor by updating its 3-tuple with the larger one. Then, the two users use the identical and synchronized CH sequence for rendezvous with other users in similar way. This process is continued until rendezvous of all users is achieved (all users eventually use the same and the largest 3-tuple).

![Fig. 6. Rendezvous of four users in three hops.](image)

We illustrate our algorithm in Fig. 6 where there are four users waiting for rendezvous. There is an edge between two users if they are within the communication range of each other. Suppose the order of 3-tuples of the four users is as follows:

$$(r_{i_0}, i_{01}, t_4) > (r_{i_0}, i_{03}, t_3) > (r_{i_0}, i_{02}, t_2) > (r_{i_0}, i_{01}, t_1).$$

Suppose rendezvous is achieved first in two pairs: user 3-user 4 and user 1-user 2. Users 3 and 1 update their 3-tuples with $(r_{i_0}, i_{01}, t_4)$ and $(r_{i_0}, i_{02}, t_2)$, respectively. Then, rendezvous is achieved in pair user 2-user 3 and user 2 updates its 3-tuple with $(r_{i_0}, i_{02}, t_2)$. Finally, rendezvous is achieved in pair user 1-user 2 again and thus rendezvous of all users is achieved. After rendezvous, all users generate the identical CH sequence by using $(r_{i_0}, i_{01}, t_1)$.

Our algorithm for rendezvous of multiple users is formally presented in JS_K as follows:

```plaintext
Algorithm JS_K
1: Input: M, C_k //for the k-th user
2: P= the smallest prime number greater than M;
3: r_0=rand[1, M]; t_0=rand[1, P]; t_0=0;
4: while (not terminated)
5: n=ceil(t/3P); r=(r_0+n−1) mod M)+1;
6: m=ceil(P/3P); l=ceil(n+t−1) mod P)+1;
7: c=JS_0p(M, P, r, l, t);
8: if c<=C_k
9: c=RandomSelect(C_k);
10: end
11: t=t+1;
12: Attempt rendezvous on channel c;//Comment 1
13: if (success in rendezvous) //Comment 2
14: h=argmax((r_{i_0}, i_{0q}, t_{i_0})); //Comment 3
15: if (r_{i_0}, i_{0q}, t_{i_0})<(r_{i_0}, i_{0q}, t_{i_0}) (r_{i_0}, i_{0q}, t_{i_0})=(r_{i_0}, i_{0q}, t_{i_0});
16: end
17: end
18: end
```

Remarks. Comment 1: exchange 3-tuple $(r_0, i_0, t)$ with its potential neighbors; Comment 2: receive 3-tuples $(r_0, i_0, t)$ from its neighbors; Comment 3: select the largest 3-tuple among all received 3-tuples.

The following two theorems present the upper bounds of MTTR of JS_K under the symmetric model and the asymmetric model, respectively.

**Theorem 3.** Under the symmetric model, $K (K \geq 2)$ users in a CRN with network diameter $D$ (in terms of the number of hops) achieve rendezvous in at most $3PD$ time slots, provided each user runs JS_K, where $P$ is the smallest prime number which is greater than $M$.

1. Given a connected network $G(V, E)$ where $V$ is a set of nodes and $E$ is a set of edges, let $h(v_i, v_j)$, $v_i, v_j \in V$, denote the minimum number of hops between nodes $v_i$ and $v_j$. The network diameter of $G$, denoted by $D$, is defined as the largest $h(v_i, v_j)$ for any $v_i, v_j \in V$, i.e., $D = \max \{h(v_i, v_j)\}_V, v_i, v_j \in V$. 
The Proof of Theorem 3 can be found in Section I in the supplemental material available online.

**Theorem 4.** Under the asymmetric model, $K (K \geq 2)$ users in a CRN with diameter $D$ (in terms of the number of hops) achieve rendezvous in at most $(3MP(P - G) + 3P)D$ time slots, provided that each user runs JS$_K$, where $P$ is the smallest prime number which is greater than $M$ and $G$ is the number of commonly available channels of all the users.

The Proof of Theorem 4 can be found in Section I in the supplemental material available online.

## 5 Simulation

We build the simulator in Matlab 7.9 to evaluate the performance of our JS algorithm. We select four representative CH algorithms, i.e., MC [12], MMC [12], DRSEQ [13], and CRSEQ [11], as baseline algorithms for comparison. These four algorithms are selected since they are recently proposed and are shown to have the overall best performance. As discussed in Section 2, MC and DRSEQ are applicable to the symmetric model, while MMC and CRSEQ can work under the asymmetric model (thus applicable to the symmetric model as well). We have presented the theoretical performance of the algorithms, in terms of upper bounds of MTTR and $E(TTR)$, in Table 1 in Section 1. In this section, we focus on the experimental results, in terms of average TTR, of the algorithms.

The simulation covers 2-user and multiuser rendezvous under both the symmetric and asymmetric models. We consider the following key parameters: $M$ (the cardinality of the whole set of potentially available channels, i.e., $M = |C|$), $K$ (the number of users involved in rendezvous), and $G$ (the number of channels commonly-available to the users). Clearly, under the symmetric model, all channels are available to any user and thus $C$ is identical to $C_i, i = 1, 2, \ldots, K$. Under the asymmetric model, $C_i \subseteq C$ and different users might have different $C_i$. We introduce another parameter $\theta (0 < \theta < 1)$ and randomly select channels from $C$ to $C_i (i = 1, 2, \ldots, K)$, such that the size of $C_i$ is equal to $\theta M$ on average.

In each simulation run, locations of cognitive users are randomly generated and uniformly distributed in a $500 \text{ m} \times 500 \text{ m}$ 2D area. All users have the same transmission radius $100 \text{ m}$. In the simulation of 2-user rendezvous, two users are assumed to be within the transmission radius of each other. In the simulation of multiuser rendezvous, we repeatedly generate random topologies of $K (K \geq 2)$ users until a connected multihop topology is found. The JS algorithm and the selected baseline algorithms are applied in the same topology of the network with the same parameter settings. Each user starts its CH sequence at a random time. TTR is counted as the number of time slots that it takes for all users to achieve rendezvous when all users have begun their sequences. The experimental results reported in the following figures are the means of 500 separate runs.

### 5.1 Two-User Rendezvous

#### 5.1.1 Under the Symmetric Model

In this scenario, we compare JS with all the baseline algorithms. Fig. 7a shows that the average TTR of all algorithms increase when $M$ increases. It is because that users can focus on less number of channels and thus increase the chance of rendezvous when $M$ is small (TTR = 1 in the extreme case $M = 1$). From Fig. 7a, we can see that performance of JS and MC is significantly better than that of other selected algorithms. Although DRSEQ is shown in Table 1a to have the smallest MTTR $2M + 1$, its average TTR is longer than that of JS and MC in the experiments (note that MTTR is the TTR in the worst cases). CRSEQ performs worst in this scenario since we have shown that MTTR of CRSEQ is at least $(P - 1)(3P - 1)$ under the symmetric model (see Table 1a in Section 1 and Lemma 3 in Appendix, available in the online supplemental material). It should be pointed out that GOS is not selected for comparison since we found in experiments that its performance is much worse than that of the baseline algorithms. According to Fig. 7a, we can see that the proposed JS algorithm can quickly conclude the rendezvous in practice. For example, when there are 50 channels, JS takes an average of 29.4 time slots, i.e., 29.4 $\times$ 20 ms = 0.588s (notice that a time slot has duration of 20ms), to conclude the rendezvous.

Performance of JS, in terms of average TTR, is very close to that of MC under the symmetric model. However, as shown in Fig. 7b, MC and MMC cannot provide guaranteed rendezvous. Note that users randomly select rates and prime numbers in MC and MMC [12]. Rendezvous fails in a trial if two users selected the identical parameters. In our simulation, users running MC/MMC are allowed to repeatedly select the parameters until the rendezvous is achieved. Fig. 7b shows the success rates with which two users achieve rendezvous successfully in a trial. Note that JS, DRSEQ, and CRSEQ have guaranteed rendezvous. The success rates of these three algorithms in a trial are always 100 percent while the success rate of MC is between 65 and 80 percent.
5.1.2 Under the Asymmetric Model

Only JS, MMC, and CRSEQ are applicable to the asymmetric model. In this scenario, we study the performance of algorithms by varying three parameters: $M$, $G$, and $θ$. We implement CRSEQ with the random-replace operation (CRSEQ-RandomReplace) in Fig. 8b. We can see that JS still performs best and achieves significant improvement over MMC and CRSEQ-RandomReplace.

Fig. 9 shows the average TTR of JS, MMC, and CRSEQ-RandomReplace against $G$ with fixed $M = 100$ and $θ = 0.8$. We exclude curve of MMC in Figs. 9b and 9d to clearly show the gap between curves of JS and CRSEQ-RandomReplace. We observe that average TTR of all algorithms decreases with increase of $G$ since more commonly available channels definitely facilitate rendezvous. Both JS and CRSEQ-RandomReplace outperform MMC significantly. JS has the best performance in terms of average TTR, especially when $G$ is relatively large, say bigger than 25. For instance, when 25 percent channels are commonly available (i.e., $25/100 = 25$ percent), the average TTR of JS is only 117 while the average TTR of CRSEQ-RandomReplace and MMC are 142 and 365, respectively (see Figs. 9c and 9d).

Figs. 10a and 10b show the success rate of JS, MMC, and CRSEQ against $M$ and $θ$, respectively. Similar to Fig. 7b, MMC cannot guarantee the rendezvous while success rate of JS and CRSEQ is 100 percent. According to Figs. 8, 9, and 10, we can conclude that JS algorithm is shown to have the overall best performance.

5.2 Multiuser Rendezvous

5.2.1 Under the Symmetric Model

Only JS is applicable to multiuser rendezvous. In order to evaluate performance of JS, we extend DRSEQ and CRSEQ by using the pairwise strategy which is adopted in our JS algorithm. Specifically, the users running DRSEQ/CRSEQ...
exchange information of their time slot counters after rendezvous. A user updates its time slot counter with the largest one among all received counters. In this way, DRSEQ and CRSEQ can be extended and thus applicable to the multiuser scenario. Note that MC and MMC cannot be extended in the similar way since users essentially generate random CH sequences in MC/MMC.

Figs. 11a, 11b, 11c, and 11d show the average TTR of JS, extended DRSEQ and CRSEQ against $M$. We study the performance of algorithms by varying $K$ from 10 to 100. Again, the average TTR of the three algorithms increases when $M$ increases and performance of JS is significantly better than other algorithms. Note that rendezvous of multiple users relies on pairwise rendezvous of two users. Superior performance of JS is accumulated in multiuser rendezvous.

Figs. 12a, 12b, 12c, and 12d show the average TTR of JS, extended DRSEQ and CRSEQ against $K$. We study the performance of algorithms by varying $M$ from 10 to 100. We find again that JS significantly outperforms the other two algorithms. From Figs. 12a, 12b, 12c, and 12d, we can see that average TTR of the three algorithms increase when $K$ increases from 5 to 40, and then decrease when $K$ further increases and is larger than 40. The reason is that more users usually require longer time for rendezvous when $K$ increases at the beginning. However, when $K$ reaches a specific value (e.g., 40 in Fig. 12), the pairwise rendezvous can be achieved simultaneously among more users and TTR is mainly dependent on the diameter of the network, in terms of number of hops. Therefore, when $K$ further increases and becomes larger than 40, the network becomes dense and the diameter of the network decreases, which subsequently decreases the average TTR.

5.2.2 Under the Asymmetric Model
Among the selected four baseline algorithms, only CRSEQ can be extended to work in this scenario. We compare JS with the extended CRSEQ in this study. Figs. 13, 14, and 15 show the average TTR of JS and CRSEQ/CRSEQ-RandomReplace against $M$, $G$, $C$, and $K$, respectively, where CRSEQ denotes its original version without using the random-replace operation and CRSEQ-RandomReplace denotes the implementation of CRSEQ with the random-replace operation (i.e., replacing
unavailable channels with available channels). According to the results shown in the figures, we can see that JS significantly outperforms CRSEQ in terms of average TTR. Similar to results in Figs. 7, 8, and 11, we can see that the average TTR of both JS and CRSEQ increase with the increase of $M$ in Fig. 13. According to Fig. 14, we can see that increase of $G$ has little impact on the average TTR in multiuser rendezvous. Increase of $G$ will definitely facilitate the average TTR of pairwise rendezvous of two users, since there are more commonly available channels between the two users. In multiuser scenario, however, when there are many users involved in the network for rendezvous, TTR of pairwise rendezvous become comparatively negligible and the diameter of the network is the dominating factor to the average TTR. Similar to the results in Fig. 12, we observe in Fig. 15 that average TTR of both algorithms increase with the slight increase of $K$ but decrease with further increase of $K$.

6 CONCLUSION

In this paper, we have studied the channel-hopping algorithms for blind rendezvous in CRNs. We consider rendezvous of 2-user and multiuser scenarios under both
the symmetric and asymmetric models. We have proposed an efficient channel-hopping algorithm which can guarantee rendezvous of $K, K \geq 2$, users in all scenarios without the need of time synchronization. We have shown the efficiency of our algorithm by the solid theoretical analysis as well as the extensive simulations.

ACKNOWLEDGMENTS

A preliminary version of this paper was presented at INFOCOM 2011 [24]. This work is supported in part by Research Grants of Hong Kong Baptist University [Project No. FRG2/10-11/111 and FRG2/11-12/160]. The corresponding author Zhiyong Lin is supported by the Natural Science Foundation of Guangdong Province in China (Grant No.: LYM11084) and the Foundation for Distinguished Young Talents in Higher Education of Guangdong in China (Grant No.: LYM11084).

REFERENCES

Hai Liu received the BSc and MSc degrees in applied mathematics from South China University of Technology, in 1999 and 2002, respectively. He received the PhD degree in computer science from City University of Hong Kong in 2006. He is currently a research assistant professor with the Department of Computer Science, Hong Kong Baptist University. His research interests include wireless networking, mobile computing, and algorithm design and analysis. He is a member of the IEEE.

Zhiyong Lin received the BSc and MSc degrees both in applied mathematics from South China University of Technology, in 1999 and 2002, respectively. He received the PhD degree in computer application technology from South China University of Technology in 2009. Currently, he is an associate professor in the Department of Computer Science, GuangDong Polytechnic Normal University. His research interests include algorithm design and analysis, machine learning and computational intelligence.

Xiaowen Chu received the BEng degree in computer science from Tsinghua University, PR China, in 1999, and the PhD degree in computer science from the Hong Kong University of Science and Technology in 2003. Currently, he is an associate professor in the Department of Computer Science, Hong Kong Baptist University. His research interests include distributed and parallel computing and wireless networks. He is a senior member of the IEEE.

Yiu-Wing Leung received the BSc and PhD degrees from the Chinese University of Hong Kong in 1989 and 1992, respectively. He has been working in the Department of Computer Science of the Hong Kong Baptist University and now he is a full professor. His research interests include two major areas: 1) networking and multimedia which include the design and optimization of wireless networks, optical networks and multimedia systems, and 2) cybernetics and systems engineering which include evolutionary computing and multi-objective programming. He has published more than 70 journal papers in these areas, and most of which were published in various IEEE journals. He is a senior member of the IEEE.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.