Channel-Hopping-Based Communication Rendezvous in Cognitive Radio Networks

Yifan Zhang, Gexin Yu, Qun Li, *Senior Member, IEEE*, Haodong Wang, *Associate Member, IEEE*, Xiaojun Zhu, *Student Member, IEEE*, and Baosheng Wang

***Abstract*—Cognitive radio (CR) networks have an ample but dynamic amount of spectrum for communications. Communication rendezvous in CR networks is the process of establishing a control channel between radios before they can communicate. Designing a communication rendezvous protocol that can take advantage of all the available spectrum at the same time is of great importance, because it alleviates load on control channels, and thus further reduces probability of collisions. In this paper, we present ETCH, efficient channel-hopping-based MAC-layer protocols for communication rendezvous in CR networks. Compared to the existing solutions, ETCH fully exploits spectrum diversity in communication rendezvous by allowing all the rendezvous channels to be utilized at the same time. We propose two protocols, SYNC-ETCH, which is a synchronous protocol assuming CR nodes can synchronize their channel hopping processes, and ASYNC-ETCH, which is an asynchronous protocol not relying on global clock synchronization. Our theoretical analysis and `ns-2`-based evaluation show that ETCH achieves better performances of time-to-rendezvous and throughput than the existing work.**

***Index Terms*—Channel hopping, cognitive radio, communication rendezvous, dynamic spectrum access.**

I. Introduction

COGNITIVE radio (CR) technology has enabled dynamic spectrum access (DSA) [2], which is a promising solution to the spectrum scarcity problem. DSA allows unlicensed users (i.e., secondary users) to access to the licensed spectrum if the spectrum is not being used by the licensed users (i.e., primary users) [3]. In CR networks, *communication rendezvous* is the first step for a pair of CR nodes (i.e., secondary users) to establish the communications with each other. In particular, a pair of CR network nodes wishing to communicate should first agree on certain control information, such as data communication channel and data rate, before they can start the data communication. The channel on which the nodes negotiate to reach the agreement is called the *control channel*. Communication rendezvous for a pair of nodes is to establish a control channel between them.

The common control channel approach, where a well-known channel is designated as the control channel for all nodes, suffers from the channel congestion problem and is vulnerable to jamming attacks [4]. Moreover, this approach cannot be applied in CR networks because the control channel itself may be occupied by the primary user and hence become unavailable to the secondary users. The channel hopping (CH) approach, by contrast, increases control channel capacity by utilizing a broader range of spectrum. With this approach, all idle nodes hop on a set of sequences of rendezvous channels (i.e., channels that are assigned for the purpose of control information exchange). When two nodes wishing to communicate hop to the same channel, this channel is utilized as the control channel between the pair of nodes. The time that it takes for a pair of nodes to establish the control channel is called " *time-to-rendezvous*, " or TTR for short.

A CH-based communication rendezvous protocol in CR networks should have the following three features. *First*, every pair of nodes should have chance to meet each other periodically with a bounded interval. *Second*, any pair of nodes should be able to meet each other on every rendezvous channel. Otherwise, a pair of nodes would not be able to communicate if the channels on which they rendezvous are occupied by the primary users, even though there are still available rendezvous channels. *Third*, all the rendezvous channels should have the same probability to be utilized as control channels. Otherwise, those heavily used rendezvous channels will have higher traffic load, and thus more collisions. Furthermore, if a CH sequence is heavily using a certain rendezvous channel, nodes hopping on this sequence will lose contact with other nodes if the heavily used channel is occupied by the primary user. Besides the above three required features, a beneficial feature of CH-based rendezvous protocols is the ability to exploit spectrum diversity, which is one of the most salient benefits offered by CR networks, in communication rendezvous. However, the existing CH-based solutions fall short on satisfying either the required or the beneficial features (Section II-B). Therefore, we propose ETCH, a set of CH-based communication rendezvous protocols that are suitable for CR networks.

ETCH contains two protocols, SYNC-ETCH and ASYN-ETCH, which target two different scenarios depending on the availability of global clock synchronization. SYNC-ETCH is a synchronous ETCH protocol that efficiently exploits the spectrum diversity in a way that *every* rendezvous

Manuscript received June 18, 2012; revised March 04, 2013; accepted May 03, 2013; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor E. Ekici. This work was supported in part by the US National Science Foundation under Grant CNS-1117412, the NSA under Grant H98230–12-1–0226, and an NSF CSUMS grant.

Y. Zhang and Q. Li are with the Department of Computer Science, College of William and Mary, Williamsburg, VA 23187-8795 USA (e-mail: yzhang@cs.wm.edu; liqun@cs.wm.edu).

G. Yu is with the Department of Mathematics, College of William and Mary, Williamsburg, VA 23185 USA (e-mail: gyu@wm.edu.

H. Wang is with the Department of Computer and Information Science, Cleveland State University, Cleveland, OH 44115 USA (e-mail: hwang@cis.csuohio.edu).

X. Zhu is with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210024, China (e-mail: gxjzhu@gmail.com).

B. Wang is with the Computer School, National University of Defence Technology (NUDT), Changsha 410073, China (e-mail: wangbaosheng@126.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2013.2270443

channel can be utilized as a control channel in *each* hopping slot. In SYNC-ETCH, while achieving the same goal, two CH sequence construction algorithms are proposed: the two-phase CH sequence construction [1] and the single-phase sequence construction. These two algorithms are complementary in design. The single-phase algorithm can *guarantee* all the rendezvous channel have the same probability to be utilized as control channels (i.e., the third required feature previously). The constraint of the single-phase algorithm is that it requires the total amount of rendezvous channels to be an odd number. The two-phase CH sequence construction algorithm can be applied to CR networks with an arbitrary number of rendezvous channel, but it tries to satisfy (but cannot guarantee) the third required feature previously. As will be shown later, both of the SYNC-ETCH CH sequence construction algorithms achieve the optimal average TTR under the premise that all the rendezvous channels should be utilized as control channels in every hopping slot. ASYNC-ETCH, on the other hand, does not assume the existence of global clock synchronization. Compared to the existing asynchronous solution [5], ASYNC-ETCH is able to take advantage of spectrum diversity by using all rendezvous channels as control channels.

The contributions of this paper are summarized as follows.

- We formulate the problem of designing channel-hopping-based communication rendezvous protocols in CR networks by considering all relevant metrics and requirements. We provide an in-depth and systematical analysis about the principles of designing this type of protocols, which is valuable for future research in this field.
- We propose an optimal synchronous protocol for communication rendezvous in CR networks. The optimality of this protocol lies in that its average time-to-rendezvous is shortest under the premise that all the rendezvous channels should be utilized in every hopping slot. This approach achieves good time-to-rendezvous while greatly increasing the capacity of the CR network at the communication setup stage.
- We propose a novel asynchronous protocol that enables two CR network nodes to rendezvous without the existence of global clock synchronization mechanisms. Our protocol achieves better time-to-rendezvous and traffic throughput than the existing schemes.

II. Related Work

The existing approaches of communication rendezvous in multichannel or CR networks can be classified into two categories.

A. Common Control Channel (CCC)-Based Approaches

The CCC-based approaches designate a commonly available channel among a group of network nodes as their control channel. Based on the method by which the common control channel is determined, the CCC-based approaches can be further divided into two classes: dedicated CCC approaches and cluster-based CCC approaches. The dedicated CCC approaches [6]–[9] assume a dedicated channel, allocated by the regulation authority, is available to all network nodes for the purpose of control information exchange.

The cluster-based CCC approaches [4], [10]–[12] relax the assumption by considering the scenario where different network nodes have different channel availability. These approaches allow nodes to self-organize into clusters based on the similarity of available channels. In each cluster, a cluster common control channel is used to carry the control information of the cluster nodes.

B. CH-Based Approaches

While the CCC-based approaches are simple in design, they are vulnerable to control channel congestion and jamming attacks [4]. The CH-based approaches [5], [13]–[17] alleviate the control channel congestion problem by utilizing a broader range of spectrum for control information exchange. Bluetooth adopts a CH method, named Adaptive Frequency Hopping (AFH), for nodes communication [14]. With AFH, all the Bluetooth devices within the same Piconet hop on the same CH sequence, which is derived from the master device address. The CH method allows Bluetooth devices to avoid interferences from other ISM band devices. However, the way that all devices adopt the same CH sequence is only ideal for the scenario where there are a small amount of concurrent communications. For CR networks with a large number of concurrent communications, this method can cause channel congestion since all the communications occur in a single channel. Therefore, the CH method adopted by Bluetooth and other similar approaches (for example, CHMA [15]) are not suitable for CR networks.

QCH [5] is a communication rendezvous protocol designed for CR networks. The synchronous version of QCH utilizes the overlap property of quorums in a quorum system to develop CH sequences such that any two CH sequences are able to rendezvous periodically. Meanwhile, to accommodate the dynamics of the channel availability in CR networks, QCH divides a period of CH sequence into several frames, where the number of the frames equals to the number of rendezvous channels. This way, two nodes following different CH sequences are guaranteed to be able to meet as long as there are rendezvous channels not being occupied. However, QCH only guarantees one channel to be utilized as control channel in each hopping slot, and thus does not satisfy the beneficial feature introduced in Section I. Furthermore, the asynchronous version of QCH only guarantees two of the rendezvous channels to be used as control channels. This arrangement also does not take advantage of spectrum diversity in CR networks, and may lead to communication outage when the primary users appear on the two channels. SeqR [16] and Jump-stay CH [17] only deal with the asynchronous scenario. Different from the previous work, we focus on exploiting the spectrum diversity, which is the most salient advantage of CR networks, in designing communication rendezvous protocols (for both synchronous and asynchronous scenarios).

III. Problem Formulation

The problem that ETCH tackles can be formulated as follows. In a CR network, there are N (orthogonal) licensed channels, labeled as $C_0, C_1, \ldots, C_{N-1}$, that can be used as control channels. Idle CR nodes (i.e., nodes waiting to initiate a communication with other nodes or nodes waiting others to connect to them) *periodically* hop on (i.e., switch their working channel according to) a *CH sequence*, which is a sequence of rendezvous channels. The time during which a node stays on a channel is

defined as a *hopping slot*, which is notated as a (slot-index, channel) pair. Thus, a CH sequence S is notated as

$$S = \{(0, S[0]), (1, S[1]), \ldots, (i, S[i]), \ldots, (p-1, S[p-1])\}$$

where $i \in [0, p-1]$ is the index of a hopping slot, and $S[i] \in \{C_0, \ldots, C_{N-1}\}$ is the rendezvous channel assigned to the ith slot of the sequence S. The time a node takes to hop through the entire CH sequence is called a *hopping period*. When two nodes hop to the same channel, they can hear from each other, and that channel is established as their control channel. If more than two nodes hop to the same rendezvous channel at the same time, they use existing collision avoidance mechanisms (e.g., RTS/CTS) or retransmission to establish pairwise control channels between them.

The design goal of ETCH is to construct CH sequences satisfying the following requirements.

- *Overlap requirement:* This requirement requires that any two CH sequences must overlap at a certain slot to ensure the rendezvous between the two nodes. Formally, given two CH sequences S_0 and S_1, they *overlap* if there exists a slot $(i, S_0[i]) \in S_0$ and a slot $(i, S_1[i]) \in S_1$ such that $S_0[i] = S_1[i]$. This slot is called an *overlapping slot* between S_0 and S_1, and the rendezvous channel $S_0[i]$ ($S_0[i] \in \{C_0, \ldots, C_{N-1}\}$) is called an *overlapping channel* between S_0 and S_1. If a rendezvous channel serves as an overlapping channel between a pair of CH sequences in the ith slot, we say that *the rendezvous channel is utilized (as a control channel) in the ith slot*.
- *Full utilization of rendezvous channels:* This requirement requires that any pair of nodes should be able to utilize every rendezvous channel as their control channel. This is to ensure the nodes can communicate with each other even if some of (but not all) the rendezvous channels are occupied by primary users.
- *Even use of rendezvous channels:* This requirement requires that all the rendezvous channels should have the same probability to appear in each CH sequence. If a CH sequence heavily relies on a certain channel (i.e., the channel is assigned to most of the slots of the CH sequence), nodes that hop on this CH sequence will lose contact with most of other nodes when the heavily relied channel is occupied by the primary user.

We use the following three metrics in our numerical analysis for the proposed ETCH scheme.

- *Average rendezvous channel load:* This metric measures the average fraction of nodes that meet in the same rendezvous channel among all the nodes. Given a CR network with M nodes and an average rendezvous channel load α ($0 < \alpha \leq 1$), there are on average $M\alpha$ nodes rendezvous in the same channel. A light rendezvous channel load alleviates traffic collisions and increases the communication throughput.
- *Average time-to-rendezvous:* This is the average number of hopping slots that two nodes need to wait before they can rendezvous. A smaller average TTR allows nodes to rendezvous and establish a communication link more quickly.
- *Rendezvous channel utilization ratio:* This is the ratio of the number of rendezvous channels that can be utilized as control channels in a hopping slot to the total number of rendezvous channels. It measures, in a given hopping slot, the extent that a communication rendezvous protocol utilizes the spectrum diversity in establishing control channels. A high rendezvous channel utilization ratio is helpful to reduce collision and improve the network capacity at the communication setup stage. This metric does not apply to the asynchronous case in which the hopping slot boundaries are not necessarily aligned.

We will show that ETCH outperforms the existing solutions through mathematical analysis and simulations in Sections IX and X, respectively.

IV. ETCH Overview

The ETCH scheme consists of two protocols: SYNC-ETCH and ASYNC-ETCH. SYNC-ETCH assumes there exists a synchronization mechanism to achieve global clock synchronization among CR nodes, while ASYNC-ETCH does not hold such an assumption.

A. Constructing CH Sequences

Before entering the CR network, a node obtains from the regulation authorities the information about the rendezvous channels used in the network (i.e., center frequencies and bandwidth of channel $C_0, \ldots, C_{N-1}$ in our problem formulation). Then, it constructs the CH sequences according to either SYNC-ETCH or ASYNC-ETCH in a distributed manner as follows.

With SYNC-ETCH (Sections V and VI), the CR node constructs a set of CH sequences by using either the *two-phase CH sequence construction* algorithm (Section V) or the *single-phase CH sequence construction* algorithm (Section VI). The two-phase algorithm can be applied to scenarios with arbitrary numbers of rendezvous channels. It satisfies the overlap requirement in the first phase and *tries* to fulfill the requirement of even use of rendezvous channels in the second phase. The single-phase algorithm *guarantees* the satisfaction of both requirements in an integral design. The key design goal of both algorithms is to fully utilize *all* the rendezvous channels in every hopping slot. Theorem 1 in the following reveals that to fully utilize all the N rendezvous channels in each hopping slot, there are at least $2N-1$ hopping slots in each CH sequence.

Theorem 1: In a CR network with N rendezvous channels, for any CH-based synchronous communication rendezvous protocols where *all* the rendezvous channels are utilized in each hopping slot, the minimum number of hopping slots of each CH sequence is $2N-1$, and the average TTR is $\frac{2N-1}{2}$.

Proof: To let all the N rendezvous channels be fully utilized in each CH time-slot, we must arrange at least $2N$ CH sequences in a way that N pairs of CH sequences rendezvous at N different channels. We also must arrange at least $2N-1$ hopping slots for each of the $2N$ CH sequences to allow each sequence to rendezvous with the rest $2N-1$ CH sequences (for the overlap requirement). Considering that the rendezvous time of two randomly selected CH sequences (from the $2N$ sequences) is uniformly distributed between slot one and slot $2N-1$, the average TTR is $\frac{2N-1}{2}$. ■

As we will show later, both CH sequence construction algorithms in SYNC-ETCH can achieve such optimal length of CH sequences.

With ASYNC-ETCH (Section VII), the CR node constructs the CH sequences in a similar fashion as SeqR [16] (both algorithms construct CH sequences using a pattern of pilot

slot leading a fixed sequence) but with a novel enhancement: ASYNC-ETCH constructs multiple CH sequences rather than only one as in SeqR. The arrangement of having multiple sequences brings two benefits. First, multiple sequences reduce the chance that two nodes select the same CH sequence. As we will show in Section VII, it takes less time for two nodes to rendezvous when they select different sequences. Second, with multiple sequences, participating nodes have more chances to rendezvous with each other within a hopping period. We show that a pair of nodes using ASYNC-ETCH that select two different CH sequences are guaranteed to rendezvous in N slots (where N is the number of rendezvous channels) within a hopping period no matter how the hopping processes of the pair of nodes are misaligned.

B. Executing CH Sequences

After constructing the CH sequences, the CR node randomly selects a sequence to execute (i.e., switches the working channel according to the randomly selected sequence). During each hopping slot, the CR node needs to sense the existence of the primary user of the slot's rendezvous channel. If the primary user is sensed, the node should yield using the channel immediately and waits until a hopping slot, in which the rendezvous channel is available, is reached before resuming the hopping process. Since our algorithm enables any pair of nodes rendezvous on different ones of the N rendezvous channels, nodes are guaranteed to be able to meet each other if there are still available channels left. The CH sequence execution process is presented in more details in Section V-D.

V. SYNC-ETCH: Two-Phase CH Sequence Construction Algorithm

A. Overview and an Example

The two-phase CH sequence construction algorithm constructs a set of CH sequences in two phases.

The first phase is called the *rendezvous scheduling* phase. In this phase, the algorithm creates a set of rendezvous schedules, each of which instruct how nodes with different CH sequences meet with each other in a hopping slot. Given a CR network with N rendezvous channels, to fully utilize spectrum diversity, an ideal rendezvous schedule allows N pairs of nodes to rendezvous at N different channels in a hopping slot, which is equivalent to arrange for $2N$ CH sequences (each of which is used by one participating node) to overlap at different N channels in a slot. Meanwhile, the rendezvous schedules should ensure the satisfaction of the overlap requirement, i.e., any pair of nodes hopping on different CH sequences can meet at least once within a hopping period.

The second phase is called the *rendezvous channel assignment* phase. In this phase, the algorithm fills the rendezvous channels in the $2N$ CH sequences based on the rendezvous schedules generated in the previous phase. This phase tries to satisfy the design requirement of even use of rendezvous channels by using a greedy algorithm. At the end of the rendezvous channel assignment phase, $2N$ CH sequences are constructed.

Figs. 1 and 2 illustrate an example of the two-phase CH sequence construction in a CR network with three ($N = 3$) rendezvous channels. Fig. 1(a) shows the five rendezvous schedules $D_0, D_1, \ldots, D_4$ generated in the rendezvous scheduling phase. Each rendezvous schedule corresponds one of the five $(2N - 1)$ hopping slots of the six $(2N)$ the CH sequences $S_0, S_1, \ldots, S_5$. As we can see, in each of the five hopping slots, six nodes (selecting different CH sequences) are supported to rendezvous in three different channels. For example, in the first slot (i.e., slot-0), where the rendezvous schedule D_0 is used to arrange rendezvous, the node selecting CH sequence S_0 is arranged to rendezvous with the node selecting CH sequence S_5 on one rendezvous channel, while the node selecting S_1 meets with the node selecting S_4 on a different rendezvous channel, and the node selecting S_2 meets with the node selecting S_3 on the remaining rendezvous channel. Fig. 1(b) shows the overall effect of how the six nodes selecting different CH sequences rendezvous in different hopping slots. In each hopping slot in Fig. 1(b), a pair of nodes whose CH sequences have the same type of shade will meet on the same rendezvous channel. Please note that the detailed arrangement about rendezvous channels on which pairs of nodes rendezvous has not yet been determined in this phase and is left to the next phase. As will be presented shortly, our algorithm schedules the $2N$ CH sequences to meet in the N different rendezvous channels in a hopping slot as follows. It selects $2N - 2$ out of the first $2N - 1$ CH sequences (i.e., $S_0, \ldots, S_{2N-2}$) to form $N - 1$ CH sequences pairs, where each sequence is scheduled to meet the other sequence from the same pair in the slot-sl, such that the index sum of each pair of CH sequences is congruent to sl modulo $2N-1$. The remaining CH sequence (within $S_0, \ldots, S_{2N-2}$) and S_{2N-1} form the last pair of CH sequences [shown in lighter color in Fig. 1(a)] that are scheduled to meet in the slot-sl.

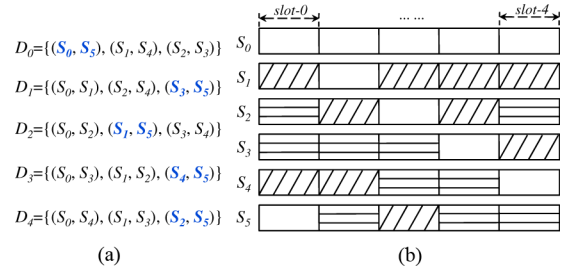

Fig. 1. Phase 1 of the two-phase CH sequence construction—*rendezvous scheduling*.

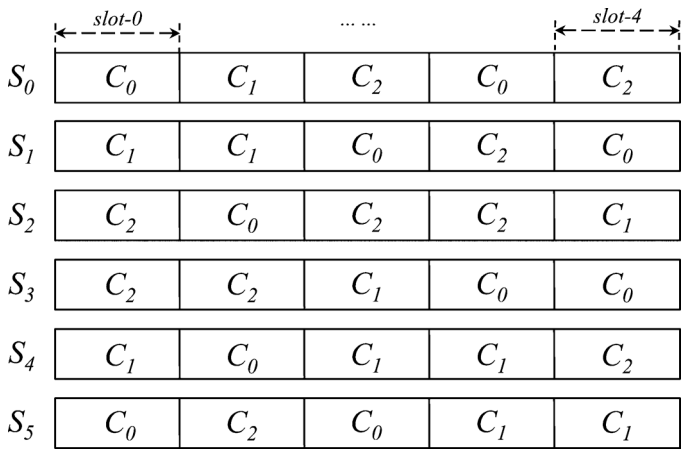

Fig. 2. Phase 2 of the two-phase CH sequence construction—*rendezvous channel assignment*.

Algorithm 1: Rendezvous Scheduling

Data: $U = \{S_0, \ldots, S_{2N-1}\}$: a set of $2N$ empty CH sequences, each of which has $2N-1$ slots;
Result: $\mathcal{D} = \{D_0, D_1, \ldots, D_{2N-2}\}$: $2N-1$ different rendezvous schedules of U.

```
Initialize D_0, D_1, ..., D_{2N-2} to be empty;
for sl ← 0 to 2N − 2 do
    V ← U\{S_{2N−2}}
    for i ← 0 to N − 1 do
        a ← the smallest subscript in V;
        If a ≤ sl the:
            b ← sl − a;
        else
            b ← 2N − 1 + sl − a;
        if a == b then
            b ← 2N − 1;
        d_i ← {S_a, S_b};
        D_sl ← D_sl ∪ {d_i};
        V ← V\{S_a, S_b};
Return D_0, D_1, ..., D_{2N−2};
```

Fig. 2 shows an example of rendezvous channel assignment once the scheduling is determined. From the example, we can see that all the rendezvous channels are utilized for communication in each of the hopping slots, and that each rendezvous channel appears in each of the CH sequences with roughly the same probability.

B. Phase 1: Rendezvous Scheduling

We now formalize the problem of rendezvous scheduling, the first phase of the two-phase CH sequence construction process, as follows. Given a set of $2N$ CH sequences $U = \{S_0, S_1 \ldots, S_{2N-1}\}$, $D_{sl} = \{d_0, d_1, \ldots, d_{N-1}\}$ is called a *rendezvous schedule* for the hopping slot indexed in sl if $\bigcup D_{sl} = d_0 \cup d_1 \cup \cdots \cup d_{N-1} = U$, where $d_i = \{S_a, S_b\}$ $(0 \le i \le N-1)$ is a pair of CH sequences that are scheduled to rendezvous in the slot-sl.

According to theorem 1, the optimal rendezvous scheduling algorithm must construct $2N-1$ different rendezvous schedules, each of which corresponds to a hopping slot, such that each CH sequence is able to rendezvous with all the other $2N-1$ CH sequences in $2N-1$ hopping slots. SYNC-ETCH uses Algorithm 1 to construct the schedules.

In Algorithm 1, rendezvous schedule D_{sl} $(0 \le sl \le 2N-2)$, which is the rendezvous schedule for the slot-sl, is constructed as follows. Within the CH sequences set $V = \{S_0, \ldots, S_{2N-2}\}$, S_a and S_b are scheduled to rendezvous in the slot-sl (i.e., $\{S_a, S_b\} \in D_{sl}$) if $a+b \equiv sl(\bmod (2N-1))$ and $a \neq b$. For the CH sequence $S_a \in V$ that satisfies $2a \equiv sl(\bmod (2N-1))$, it is scheduled to rendezvous with the CH sequence S_{2N-1} in the slot-sl (i.e., $\{S_a, S_{2N-1}\} \in D_{sl}$).

Due to the space limit, we present the proof of the correctness of Algorithm 1 in the full version of the paper [18].

C. Phase 2: Rendezvous Channel Assignment

In the second phase of the two-phase algorithm, we assign rendezvous channels to each of the $2N$ CH sequences according to the rendezvous schedules generated in the previous phase. The goal of the rendezvous channel assignment phase is twofold. *First*, to fully exploit the frequency diversity of a CR network in establishing control channels, *all* the rendezvous channels should be utilized in each hopping slot. *Second*, the assignment tries to satisfy the even use of rendezvous channels requirement presented in Section III by an arrangement that allows each rendezvous channel to have a roughly equal probability to appear in each CH sequence.

We employ a heuristic greedy algorithm to achieve the goals of rendezvous channel assignment. The details of the algorithm are presented in the full version of the paper [18] and are omitted here due to the space limit.

Fig. 2 shows the result of rendezvous channel assignment in a CR network with three rendezvous channels, C_0, C_1, and C_2. CH sequences S_0 to S_5 are the final CH sequences constructed by the two-phase CH sequence algorithm.

D. CH Sequence Execution

After constructing CH sequences, the newly joined node obtains a set of CH sequences, which are the same as those that any other nodes construct. Then, the node synchronizes to the existing nodes using the global synchronization mechanism and starts the channel hopping process described as follows. The node randomly selects a CH sequence to hop on. After hopping through all the slots, it performs the random CH sequence selection again and starts hopping on the newly chosen CH sequence. The node repeats this process while it is idle. The reason for the node to reselect a CH sequence after a hopping period is to make sure that any pair of nodes are able to rendezvous in different rendezvous channels. Since the selection of CH sequence is random, the requirement of full utilization of rendezvous channels is satisfied. When a rendezvous channel's primary user appears, the nodes on that channel should yield using the channel, wait until a hopping slot, in which the rendezvous channel is available, is reached, and resume the hopping process.

The CH sequence execution process above also applies to the case when CH sequences are constructed by either SYNC-ETCH's single-phase algorithm (Section VI) or by ASYNC-ETCH (Section VII) (a small difference with ASYN-ETCH is that nodes do not need to perform clock synchronization before starting the hopping process). Therefore, we will omit the CH sequence execution description in those two sections later.

VI. SYNC-ETCH: Single-Phase CH Sequence Construction Algorithm

A. Overview and an Example

In a CR network with N rendezvous channels, similar to the two-phase algorithm, the single-phase CH sequence construction algorithm constructs $2N$ CH sequences, each of which has $2N-1$ hopping slots, such that the following two requirements are satisfied. *First*, every CH sequence meets with all the other $2N-1$ CH sequences each at a time in a hopping slot. *Second*, all the rendezvous channels are utilized for CH sequence rendezvous in every hopping slot. The improvement of the single-phase algorithm over the two-phase algorithm is that it can *guarantee* to satisfy a *third* requirement that every rendezvous channel has the same probability to appear in each CH sequence (i.e., the even use of rendezvous channels requirement). For instance, in a CR network with three rendezvous

channels, the even use of rendezvous channels requirement expects that, within the five hopping slots of *each* CH sequence, there are two rendezvous channels appearing twice, and the remaining channel appears once. However, in the six CH sequences constructed by the two-phase algorithm (shown in Fig. 2), channels C_2 and C_1 appear three times in the CH sequence S_2 and S_4 respectively. By contrast, the single-phase algorithm can guarantee the even use of rendezvous channels requirement.

The single-phase algorithm views the rendezvous among the CH sequences within a hopping period as a colored graph G. To satisfy the three requirements above, the colored graph G should have the following properties.

First, there are $2N$ vertices in G (each vertex corresponds to one of the $2N$ CH sequences)

$$|V(G)| = 2N. \tag{1}$$

Second, the edges in G have N different colors (each color corresponds to a rendezvous channel):

$$C(G) = \{c_0, \ldots, c_{N-1}\}. \tag{2}$$

Third, since every CH sequence should rendezvous with all the other $2N-1$ sequences exactly once in a hopping period, the graph G should satisfy

$$\forall v \in V(G), d(v) = 2N - 1. \tag{3}$$

Fourth, since all the rendezvous channels should appear in every CH sequence, the color degree of each vertex, which is the number of colors on the edges incident to the vertex, should be N

$$\forall v \in V(G), \delta(v) = N. \tag{4}$$

Fifth, since each of the N rendezvous channels should have the same probability to appear in every CH sequence, among the N different colors on the $2N-1$ edges incident to a vertex v, there should be one color to appear once and the remaining $N-1$ colors to appear twice

$$\forall v \in V(G)\ (\exists c_i \in C(G)\ (\delta_{c_i}(v) = 1 \quad \delta_{c_j}(v) = 2, \forall j \in [0, N-1], j \neq i)) \tag{5}$$

where $\delta_{c_i}(v)$ is the number of edges colored with c_i that are incident to the vertex v.

For example, Fig. 3(a) shows the graph G for a CR network with five rendezvous channels (i.e., $N = 5$). The graph G in the example is a five-colored 10-vertex complete graph K_{10}. In this graph, each of the five colors has an even probability to appear on the nine edges that are incident to each vertex (i.e., one color appears once, and each of the rest four colors appear twice), which is the best case of satisfying the even use of rendezvous channels requirement.

The graph G with the properties (1)–(5) tells how each of the $2N$ CH sequence meets with each other in the N rendezvous channels *within a whole hopping period*. The single-phase algorithm needs to further specify how the CH sequences rendezvous with each other in each of the $2N-1$ hopping slots. To fully exploit the spectrum diversity, our algorithm ensures that all the rendezvous channels can be utilized as control channel in every hopping slot. This is achieved by decomposing the

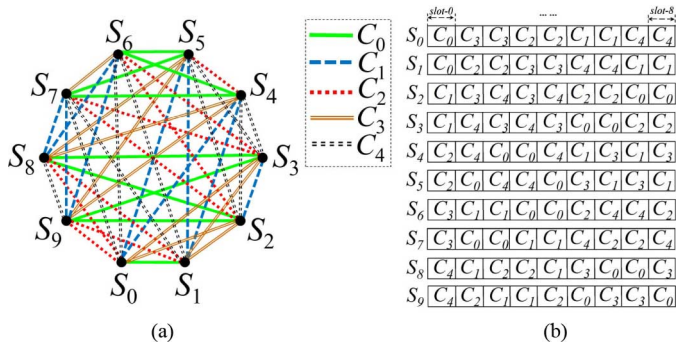

Fig. 3. For a CR network with five rendezvous channels (i.e., $N = 5$), (a) is the N-colored $2N$-vertex complete graph G that shows how the $2N$ CH sequence rendezvous with each other within a hopping period, and (b) shows the final $2N$ CH sequences.

graph G into $2N-1$ *different perfect rainbow matchings*, each of which instructs how the $2N$ CH sequences rendezvous in a hopping slot. In graph theory, a *matching* in a graph G is a set of edges of G without common vertices, a *perfect matching* in G is a matching that covers all the vertices of the graph G[19], and a *rainbow matching* in G is a matching where edges have distinct colors [20]. Therefore, in our case, a *perfect rainbow matching* (notated as PRM) in an N-colored $2N$-vertex complete graph G is an edge set that contains N disjoint edges of G colored with the N distinct colors

$$\begin{aligned}\mathrm{PRM} = \{\hat{E} \mid\ & V(\hat{E}) = V(G) \\ & \forall e_i \in \hat{E}\ (\forall v \in V(e_i)\ (v \notin V(e_j) \\ & \forall e_j \in \hat{E}, j \neq i)) \forall e_i, e_j \in \hat{E},\ i \neq j\ (C(e_i) \neq C(e_j))\}\end{aligned} \tag{6}$$

where $V(\hat{E})$ denotes the set of vertices of the edge set $\hat{E}$, $V(e_i)$ denotes the two vertices on the edge e_i, and $C(e_i)$ denotes the color on the edge e_i. Given two perfect rainbow matching PRM_i and PRM_j, they are different if and only if the there are no common edges in them

$$\mathrm{PRM}_i \text{ and } \mathrm{PRM}_j \text{ are different } \texttt{iff} \quad \forall e_i \in E(\mathrm{PRM}_i), e_j \in E(\mathrm{PRM}_j)\ (e_i \neq e_j) \tag{7}$$

where $E(\mathrm{PRM}_i)$ denotes the edge set in the perfect rainbow matching PRM_i. In our example, the five-colored 10-vertex complete graph G shown in Fig. 3(a) can be decomposed into nine different PRM's shown in Fig. 4(1), (6a-1)–(6a-4), and (6b-1)–(6b-4), respectively. Each of these PRM's is the rendezvous schedule for one hopping slot within a hopping period. The final CH sequences [shown in Fig. 3(b)] are constructed based on these nine different PRM's.

B. Intuitive Description of the Algorithm

In this section, we show that for a CR network with N rendezvous channels, where N is an odd number greater than two, our single-phase CH sequence construction algorithm can form a graph G with the properties of (1)–(5) and decompose the graph G into $2N-1$ different perfect rainbow matchings [i.e., properties of (6) and (7)].

Initial State: Initially, the graph G only contains the $2N$ vertices and no edges. Our goal is to add colored edges to G to make it an N-colored $2N$-vertex complete graph K_{2N} with the

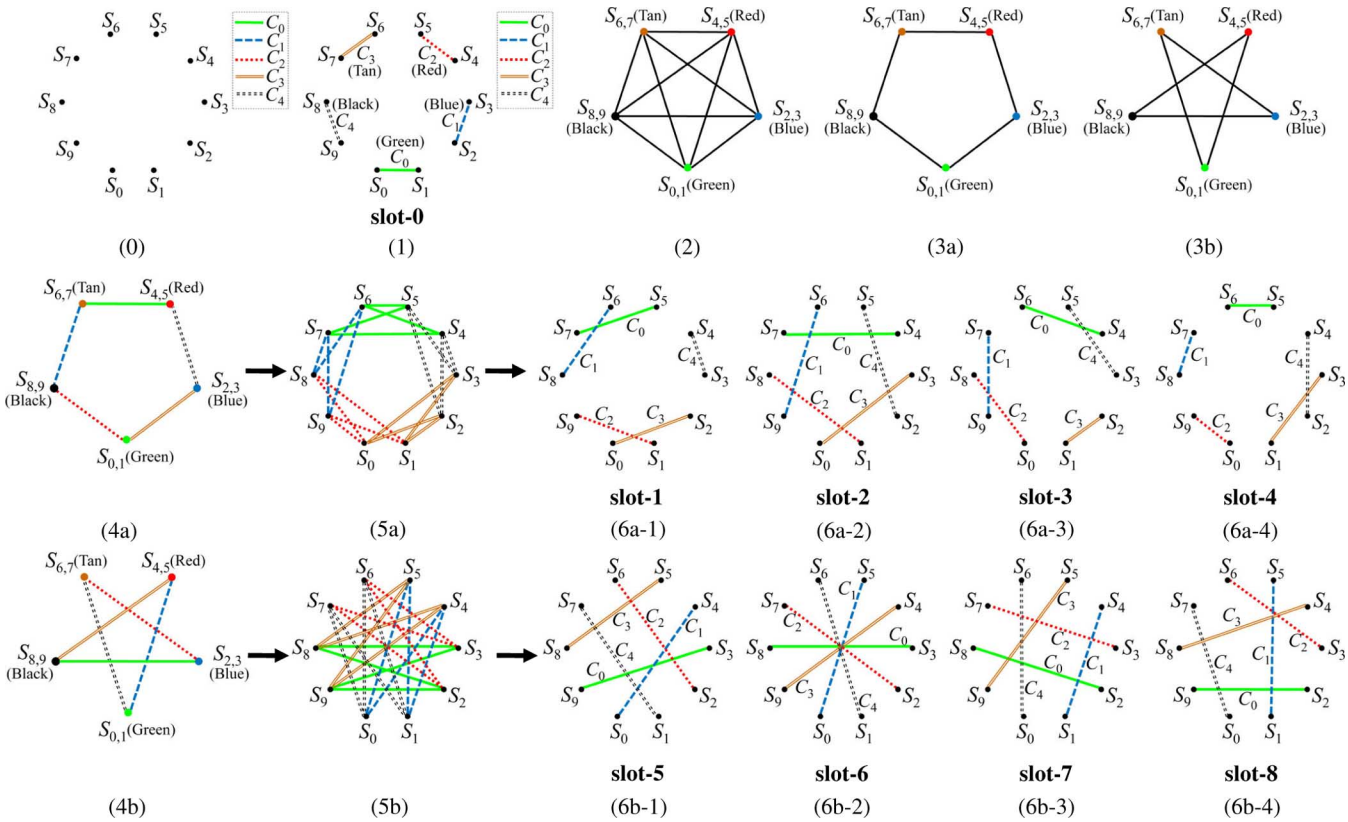

Fig. 4. Six steps of the intuitive description of the single-phase CH sequence construction algorithm for a CR network with five rendezvous channels (i.e., $N = 5$). (0) is the initial state of the graph G. (1) shows how the first PRM is formed in step #1. (2) shows how the graph G is shrunk to K_5 in step #2 based on the first PRM. (3a) and (3b) are the two 2-factors of K_5 obtained in step #3. (4a) and (4b) show the rainbow-coloring of the two 2-factors of K_5 in step #4. (5a) and (5b) show how the two rainbow-colored 2-factors of K_5 are expanded back to the two 4-factors in K_{10} in step #5. (6a-1)–(6a-4) and (6b-1)–(6b-4) are respectively the final 4 PRM's decomposed from the two rainbow-colored 4-factors of K_{10}.

properties (1)–(5), and then decompose the K_{2N} into $2N-1$ different perfect rainbow matchings [i.e., properties (6) and (7)]. In the following, we notate the $2N$ CH sequences as $S_0 \cdots S_{2N-1}$, and the N rendezvous channels as $C_0, \ldots, C_{N-1}$. We will take a CR networks with five rendezvous channels (i.e., $N = 5$) as an example throughout this section. Fig. 4(0) shows the initial state of the graph G in the example.

Step #1—Forming the First Perfect Rainbow Matching of K_{2N}: In the first step, we form the first PRM by connecting vertex S_i with vertex S_{i+1} by using an edge with the color of channel $C_{\frac{i}{2}}$, where i is an even number in the range of $[0, 2N-1]$. This PRM tells how the $2N$ CH sequences rendezvous with each other in the hopping slot-0. In our example, Fig. 4(1) shows the state of the graph G after step #1 is applied. It is a PRM of a $2N$-vertex complete graph K_{2N}, which specifies that in the hopping slot-0, CH sequences S_0 and S_1, S_2 and S_3, S_4 and S_5, S_6 and S_7, S_7 and S_8 rendezvous in channel C_0 to C_4, respectively.

Step #2—Shrinking the $2N$ Vertex Graph G to K_N: In the second step, we shrink the $2N$-vertex graph G to an N-vertex N-colored complete graph K_N as follows. First, we combine every connected vertex pair in the first PRM of K_{2N} (i.e., $\{S_i, S_{i+1}\}$, where $i = 2a, a \in [0, N-1]$) into a new vertex (notated as $S_{i,i+1}$) and connect the new vertices to each other to form a N-vertex complete graph K_N. Second, we give each vertex in K_N the color of the edge connecting the corresponding vertex pair in the first PRM of K_{2N}. Fig. 4(2) shows the state of the graph G after the step #2 is applied.

Step #3—Decomposing K_N Into $\frac{N-1}{2}$ Different 2-Factors: In graph theory, a 2-factor of a graph G is spanning subgraph of G, where the degree of each vertex in the subgraph is 2. Algorithm 2 decomposes K_N into $\frac{N-1}{2}$ 2-factors. In the algorithm, each edge of K_N and its vertices are put into $\frac{N-1}{2}$ graphs, $TF_1, \ldots, TF_{\frac{N-1}{2}}$, depending on the subscript difference between the two vertices: given an edge $e = (S_{i,i+1}, S_{j,j+1}) \in E(K_N)$, where $i < j$, the edge e and its vertices are put into the graph TF_d $(1 \le d \le \frac{N-1}{2})$ if either $\frac{j-i}{2}$ or $\frac{i+2N-j}{2}$ equals to d (line 5). Each resulting graph TF_d $(1 \le d \le \frac{N-1}{2})$ is a 2-factor of the complete graph K_N (the proof is given in the full version of the paper [18]). More specifically, TF_d is either a Hamiltonian cycle of K_N if d and N is coprime, or $\mathrm{GCD}(N, d)$ disjoint $\frac{N}{\mathrm{GCD}(N,d)}$-cycles (i.e., cycles with $\frac{N}{\mathrm{GCD}(N,d)}$ edges), where $\mathrm{GCD}(N, d)$ is the greatest common divisor of N and d, otherwise. It is obvious that the $\frac{N-1}{2}$ 2-factors are different, since each edge of K_N can only belong to one 2-factor.

In our example, Fig. 4(3a) and (3b) are the two 2-factors of the complete graph K_5 obtained after the step #2. They are both Hamiltonian cycles of K_5.

Step #4—Rainbow-Coloring 2-Factors of K_N: In the fourth step, we color each of K_N's 2-factors using all the N colors such that each edge will have a different color, which is a process called "rainbow-coloring."

Algorithm 2: 2-factorization of complete graph K_N

Data: K_N formed after step #2, the N vertices of K_N are $\{S_{0,1}, \ldots, S_{2N-2,2N-1}\}$;
Result: $\frac{N-1}{2}$2-factors of K_N: $TF_1, \ldots, TF_{\frac{N-1}{2}}$.

for $d \leftarrow 1$ **to** $\frac{N-1}{2}$;**do**
  $TF_d \leftarrow \phi$;
  for *each edge* $(S_{i,i+1}, S_{j,j+1}) \in E(K_N)$**do**
    If $\frac{|i-j|\%N}{2} == d$**then**
      Add edge $(S_{i,i+1}, S_{j,j+1})$ and its vertices to TF_d;
Return $TF_1, \ldots, TF_{\frac{N-1}{2}}$;

Algorithm 3: Rainbow-coloring a 2-factor of K_N

Data: A 2-factor TF obtained in step #3;
Result:: Rainbow-coloring the 2-factor such that every edge's color is different from the colors of its two vertices.

for *each edge* $e = (S_{i,i+1}, S_{j,j+1}) \in TF$ **do**
  If $\frac{i+j}{2}$ is an even number **then**
    Put the color of vertex $S_{\frac{i+j}{2}, \frac{i+j}{2}+1}$ on the edge e;
  else
    Put the color of vertex $S_{\frac{i+j+2N}{2}\%2N, \frac{i+j+2N}{2}\%2N+1}$ on the edge e;

Algorithm 3 shows the rainbow-coloring algorithm. In this algorithm, the color to put on a edge is the color of another vertex that is different from the two vertices of the edge. Specifically, the color to put on the edge $(S_{i,i+1}, S_{j,j+1})$ is the color of either the vertex $S_{\frac{i+j}{2}, \frac{i+j}{2}+1}$ (if $\frac{i+j}{2}$ is an even number) or the vertex $S_{\frac{i+j+2N}{2}\%2N, \frac{i+j+2N}{2}\%2N+1}$ (if $\frac{i+j}{2}$ is an odd number). After the coloring process, the two 2-factors of K_N have the following two properties.

- The colors on the N edges of each 2-factor are different (i.e., it is a rainbow-coloring).
- Second, putting the $\frac{N-1}{2}$ 2-factor together (which forms the complete graph K_N), the colors on the $N-1$ edges incident to a vertex $S_{i,i+1}$ are different; these $N-1$ colors are also different from the color of the vertex $S_{i,i+1}$.

Due to the space limit, the proofs of the above two properties are given in the full version of the paper [18].

In our example, Fig. 4(4a) and (4b) show the rainbow-coloring for the two 2-factors of K_5.

Step #5—Expanding the Rainbow-Colored 2-Factors of K_N Back to 4-Factors of K_{2N}: Here, we expand each 2-factor of K_N back to a 4-factor of the $2N$-vertex complete graph K_{2N}. For each edge $e = (S_{i,i+1}, S_{j,j+1})$ in a 2-factor of K_N, we expand it to a *monochromatic complete bipartite graph* $K_{2,2}$

$$\mathrm{MCB}_{i,j} = (V_i + V_j, E_{ij}) \tag{8}$$

where V_i and V_j are the vertex pairs $\{S_i, S_{i+1}\}$ and $\{S_j, S_{j+1}\}$ in the original $2N$-vertex graph G, and E_{ij} is the edge set of $K_{2,2}$. Additionally, we give all the four edges in E_{ij} the same color as that on the edge $e = (S_{i,i+1}, S_{j,j+1})$ in K_N. After this process, every 2-factor in K_N is expanded to a 4-factor in K_{2N}. Since the $\frac{N-1}{2}$ 2-factors of K_N are different, we have obtained $\frac{N-1}{2}$ different 4-factors of K_{2N}.

1: $(\lambda_0[0], \lambda_1[0]), (\lambda_1[1], \lambda_2[0]), (\lambda_2[1], \lambda_0[1])$
2: $(\lambda_0[0], \lambda_1[1]), (\lambda_1[0], \lambda_2[1]), (\lambda_2[0], \lambda_0[1])$
3: $(\lambda_0[1], \lambda_1[0]), (\lambda_1[1], \lambda_2[1]), (\lambda_2[0], \lambda_0[0])$
4: $(\lambda_0[1], \lambda_1[1]), (\lambda_1[0], \lambda_2[0]), (\lambda_2[1], \lambda_0[0])$

Fig. 5. Dividing the edges of a CMCB that contains three MCB's, i.e., $\mathrm{CMCB} = \{\lambda_0, \lambda_1, \lambda_2\}$, into four groups such that all the edges in each group have no common vertex.

The $\frac{N-1}{2}$ different 4-factors of K_{2N} together with the first PRM obtained in the step #2 form the N-colored $2N$-vertex complete graph K_{2N} that has the properties of (1) to (5) (the formal proof is shown in the full version of the paper [18]).

In our example, Fig. 4(5a) and (5b) show the two 4-factors of K_{10} that are converted from the two 2-factors of K_5. These two 4-factors and the first PRM obtained in step #2 [i.e., Fig. 4(2)] together form the complete graph K_{10} with the properties of (1)–(5) [Fig. 3(a)].

Step #6—Decomposing the 4-Factors of K_{2N} Into Perfect Rainbow Matchings in K_{2N}: Finally, we decompose each of the 4-factors of K_{2N} obtained in step #5 into four different PRM's such that the properties (6) and (7) are satisfied.

In the previous step, each edge $(S_{i,i+1}, S_{j,j+1})$ in a 2-factor TF_d ($d \in [1, \frac{N-1}{2}]$) of K_N is expanded to a monochromatic complete bipartite graph $\mathrm{MCB}_{i,j}$. Furthermore, recall that TF_d is either a Hamiltonian cycle of K_N (when $\mathrm{GCD}(N,d) = 1$) or a set of $\mathrm{GCD}(N,d)$ disjoint $\frac{N}{\mathrm{GCD}(N,d)}$-cycles (when $\mathrm{GCD}(N,d) \neq 1$), where $\mathrm{GCD}(N,d)$ is the greatest common divisor of N and d. Therefore, the 4-factor FF_d of K_{2N}, which is expanded from the 2-factor TF_d of K_N, is a spanning graph of K_{2N} consisting of either one *chained monochromatic complete bipartite* graph (notated as CMCB) (when $\mathrm{GCD}(N,d) = 1$) or a set of $\mathrm{GCD}(N,d)$ disjoint CMCB's (when $\mathrm{GCD}(N,d) \neq 1$).

Since a monochromatic complete bipartite graph $\mathrm{MCB}_{i,j}$ connects two pairs of unconnected vertices $\{S_i, S_{i+1}\}$ and $\{S_j, S_{j+1}\}$, each CMCB in the 4-factor FF_d ($d \in [1, \frac{N-1}{2}]$) of K_{2N} can be expressed as

$$< \lambda_0, \lambda_1, \ldots, \lambda_{n-1}, \lambda_0 > \tag{9}$$

where $n = \frac{N}{\mathrm{GCD}(N,d)}$ is the number of MCB's contained in the CMCB, and λ_p ($p \in [0, n-1]$) is the pth unconnected vertex pair in the CMCB: $\{S_{2pd\%2N}, S_{(2pd+1)\%2N}\}$. For example, for the first 4-factor FF_1 of K_{10} shown in Fig. 4(5a), we have $\lambda_0 = \{S_0, S_1\}$, $\lambda_1 = \{S_2, S_3\}$, $\lambda_2 = \{S_4, S_5\}$, $\lambda_3 = \{S_6, S_7\}$, and $\lambda_4 = \{S_8, S_9\}$. Meanwhile, for the second 4-factor FF_2 of K_{10} shown in Fig. 4(5b), we have $\lambda_0 = \{S_0, S_1\}$, $\lambda_1 = \{S_4, S_5\}$, $\lambda_2 = \{S_8, S_9\}$, $\lambda_3 = \{S_2, S_3\}$, and $\lambda_4 = \{S_6, S_7\}$.

Given a CMCB in a 4-factor expressed in (9), we divide the edges of the CMCB into four groups as follows. For the four edges of each MCB, we put them into four groups, respectively, such that edges in the same group share no common vertex. Figs. 5 and 6 show the way we divide the edges. Fig. 5 shows the case that the CMCB has three MCB's, and Fig. 6 shows the case that the CMCB has more than three MCB's. In these two figures, $\lambda_p[0]$ and $\lambda_p[1]$ are the first and the second vertex of the vertex pair λ_p ($p \in [0, 1]$), respectively.

If a 4-factor of K_{2N} contains one CMCB (when $\mathrm{GCD}(N,d) = 1$), the four edge groups obtained by using the dividing method shown in Fig. 6 are the four different

1: $(\lambda_0[0],\lambda_1[0]),(\lambda_1[1],\lambda_2[0]),(\lambda_2[1],\lambda_3[1]),$ $(\lambda_3[0],\lambda_4[0]),(\lambda_4[1],\lambda_5[1]),\ldots,(\lambda_{n-2}[0],\lambda_{n-1}[0]),(\lambda_{n-1}[1],\lambda_0[1])$
2: $(\lambda_0[0],\lambda_1[1]),(\lambda_1[0],\lambda_2[1]),(\lambda_2[0],\lambda_3[1]),$ $(\lambda_3[0],\lambda_4[1]),(\lambda_4[0],\lambda_5[1]),\ldots,(\lambda_{n-2}[0],\lambda_{n-1}[1]),(\lambda_{n-1}[0],\lambda_0[1])$
3: $(\lambda_0[1],\lambda_1[0]),(\lambda_1[1],\lambda_2[1]),(\lambda_2[0],\lambda_3[0]),$ $(\lambda_3[1],\lambda_4[1]),(\lambda_4[0],\lambda_5[0]),\ldots,(\lambda_{n-2}[1],\lambda_{n-1}[1]),(\lambda_{n-1}[0],\lambda_0[0])$
4: $(\lambda_0[1],\lambda_1[1]),(\lambda_1[0],\lambda_2[0]),(\lambda_2[1],\lambda_3[0]),$ $(\lambda_3[1],\lambda_4[0]),(\lambda_4[1],\lambda_5[0]),\ldots,(\lambda_{n-2}[1],\lambda_{n-1}[0]),(\lambda_{n-1}[1],\lambda_0[0])$

Fig. 6. Dividing the edges of a CMCB $= \{\lambda_0,\ldots,\lambda_{n-1},\lambda_0\}$, where n is the number of MCB's in the CMCB and n is greater than 3, into four groups such that all the edges in each group have no common vertex.

PRM's. If the 4-factor contains several disjoint CMCB's (when $\mathrm{GCD}(N,d) \neq 1$), we put the ith $(1 \leq i \leq 4)$ edge group of each CMCB into the same group to form a PRM. Therefore, each 4-factor of K_{2N} leads to four different PRM's, and the $\frac{N-1}{2}$ different 4-factors of K_{2N} produce $2N-2$ different PRM's. Adding the first PRM obtained in step #1, we now have $2N-1$ different PRM's of K_{2N}. Each of these PRM's instructs the $2N$ CH sequences rendezvous in one of the $2N-1$ hopping slots of a hopping period.

Since the edges in the same PRM share no common vertex and the colors of the K_{2N}'s MCBs are different, the property (6) is satisfied. Meanwhile, since each edge is assigned to only one PRM, the $2N-1$ PRM's are different (i.e., the property (7) is satisfied).

In our 5-rendezvous channel network example (i.e., $N=5$), using the dividing method in Fig. 6, the first 4-factor of K_{10} [Fig. 4(5a)] is decomposed into four different PRM's shown in Fig. 4(6a-1)–(6a-4), and the second 4-factor of K_{10} [Fig. 4(5b)] is decomposed into another four different PRM's shown in Fig. 4(6b-1)–(6b-4). Based on the nine PRM's [i.e., Fig. 4(2), (6a-1)–(6a-4), and (6b-1)–(6b-4)], we construct the final $2N$ CH sequences shown in Fig. 3(b).

C. Complete Algorithm

Due the space limit, the complete algorithm is shown in the full version of the paper [18]. It is essentially a summary of the intuitive description of the algorithm described previously.

VII. ASYNC-ETCH

SYNC-ETCH cannot guarantee channel overlap for rendezvous without the assumption of global clock synchronization. We develop an asynchronous scheme, ASYNC-ETCH, to address the issue.

A. Overview and An Example

Fig. 7 shows an example of the CH sequences constructed by SYNC-ETCH. In the example, there are $N=5$ rendezvous channel in the network. The nodes construct a set of four (i.e., $N-1$) CH sequences, S_0, S_1, S_2, and S_3. Each CH sequence consists of five (i.e., N) frames, each of which contains 11 (i.e., $2N+1$) slots: a pilot slot followed by two 5-slot (i.e., N-slot) subsequences. The channel orders of the pilot slots in S_0 to S_3 are shown as the sequences of A_0 to A_3 in the upper left part of the figure. The arrangements for A_0 to A_3 are derived by the method of addition modulo the prime number 5 (i.e., N) with different addends from one to four, respectively. The construction of the four subsequences (shown in the upper right part of the figure) also follows the channel assignment order determined in A_0 to A_3. As we will prove later, the above CH sequence construction guarantees that any pair of nodes (selecting two different sequences) rendezvous in N slots within a hopping period regardless how much channel hopping misalignment between the two nodes. Each ASYNC-ETCH CH sequence has 55 (i.e., $N*(2N+1)$) slots.

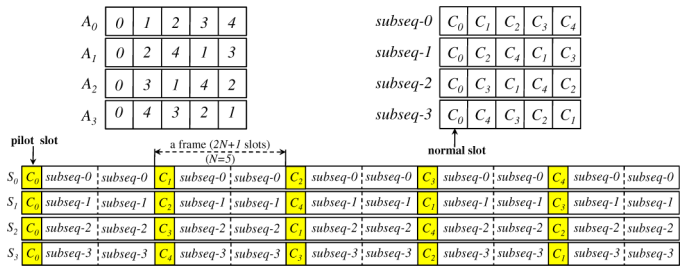

Fig. 7. CH sequences of a CR network with five rendezvous channels.

Algorithm 4: Async. CH sequence Construction

Data: $\mathcal{C} = \{C_0,\ldots,C_{N-1}\}$: N rendezvous channels (N is prime).
Result: $S_0,\ldots,S_{N-2}$: $N-1$ final CH sequences.

```
1  for i ← 0 to N − 2 do
2      A_i[0] ← 0;
3      for j ← 1 to N − 1 do
4          A_i[j] ← (A_i[0] + j(i + 1)) mod N;
5  for i ← 0 to N − 2 do
6      for j ← 0 to N − 1do
7          subSeq_i[j] ← C_{A_i[j]};
8  for i ← 0 to N − 2 do
9      k ← 0;
10     for j ← 0 to 2N^2 + N − 1 do
11         If j mod (2N + 1) == 0 then
12             S_i ← S_i ∪ (j, subSeq_i[j/(2N+1)]); // pilot slot
13         else
14             S_i ← S_i ∪ (j, subSeq_i[k]); // normal slot
15             k ← (k + 1) mod N;
16 Return S_0, S_1, ..., S_{N−2};
```

B. CH Sequences Construction

Algorithm 4 describes the construction of the $N-1$ CH sequences in ASYNC-ETCH. To ease our presentation, we assume the number of rendezvous channels, N, is a prime number. We hold the discussion of a general case (where N is not prime) until Section VIII. Given N rendezvous channels, ASYNC-ETCH first derives $N-1$ integer sequences A_0 through A_{N-2} (which will be used as indices for later channel assignment) by applying addition modulo the prime number N (lines 1–4). Note that all the integer sequences are derived with different addends. In lines 5–7, the algorithm constructs $N-1$ CH subsequences, $subSeq_0$ to $subSeq_{N-2}$, whose channel

indices are the same as the integer sequences A_0 through A_{N-2} respectively. Next, the algorithm constructs the CH sequence S_i $(0 \leq i \leq N-2)$ by concatenating five frames of S_i together (lines 8–15). Each frame of S_i consists of a *pilot slot* followed by a pair of $subSeq_i$. Slots in $subSeq_i$ are referred as *normal slots*. The channels in S_i's pilot slots, combined together, are exactly channels appearing in $subSeq_i$ in the same order. From Algorithm 4, it is easy to see that ASYNC-ETCH fulfills the requirement of even use of the rendezvous channels.

C. Proof of Rendezvous

In ASYNC-ETCH, the TTR between a pair of nodes is related to the fact that whether the two nodes select the same CH sequence or two different ones. Here, we provide the theoretical analysis to determine the TTR performance in the above two situations. In particular, we prove that the two nodes have at least one overlapped CH slot within a hopping period in the former case, and they can rendezvous at least N times in the latter one. Let us first rewrite the definition of *rotation closure property* from QCH [5] as follows.

Definition 1: Given a CH sequence S with p slots and a non-negative integer d, $\mathcal{R}(S,d) = \{(i, \mathcal{R}(S,d)[i]) \mid \mathcal{R}(S,d)[i] = S[(i+d) \bmod p]\}$ is called *a rotation of S with distance d*.

Definition 2: A CH sequence S with p slots is said to have the *rotation closure property with a degree of overlapping m* if $\forall d \in [0, p-1]$, $|S \cap \mathcal{R}(S,d)| \geq m$.

For instance, considering a CH sequence with three hopping slots, $S = \{(0,C_0),(1,C_0),(2,C_1)\}$, the two possible rotations are $\mathcal{R}(S,1) = \{(0,C_0),(1,C_1),(2,C_0)\}$ and $\mathcal{R}(S,2) = \{(0,C_1),(1,C_0),(2,C_0)\}$. It is obvious that S has the rotation closure property with a degree of overlapping 1.

Different from the prior work in SeqR [16], ASYNC-ETCH constructs multiple CH sequence rather than a single one. We provide the following definition to distinguish one CH sequence from another.

Definition 3: Two CH sequences, S_0 and S_1, each with p slots, are said be *different* if $\forall d \in [0, p-1]$, $S_1 \neq \mathcal{R}(S_0, d)$.

It is obvious that the $N-1$ CH sequences constructed by Algorithm 4 are different since the subsequences, which are the building blocks of the CH sequences, are different.

We first analyze the case that two nodes select the same CH sequence.

Lemma 1: For two nodes periodically hopping on a CH sequence that has the closure property with a degree of overlapping m, they can rendezvous in at least $\frac{m}{2}$ slots within a hopping period no matter how their hopping processes are misaligned.

Proof: This lemma has been proved in QCH [5]. ■

Theorem 2: For two nodes that select the same CH sequence constructed by Algorithm 4, they can rendezvous in at least 1 slot within a hopping period no matter how their hopping processes are misaligned.

Proof: Presented in the full version of the paper [18]. ■

To determine the rendezvous performance when two nodes select two different CH sequences, we first give the definition of integer sequences derived by the method of addition modulo a prime number with different addends and prove its overlap property.

Definition 4: Two integer sequences, $A = \{a_0, \ldots, a_{N-1}\}$ and $B = \{b_0, \ldots, b_{N-1}\}$ where N is a prime number, are said to be derived by the method of addition modulo the prime number N with different addends m and n if $a_i = (a_0 + im) \bmod N$, $b_i = (b_0 + in) \bmod N$, where $0 \leq a_0, b_0 \leq N-1$, $1 \leq i \leq N-1$ and $1 \leq m \neq n \leq N-1$.

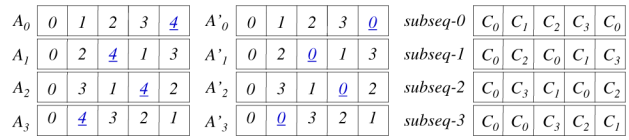

Fig. 8. ASYNC-ETCH CH sequences construction in a CR network with four rendezvous channels.

Lemma 2: Given two integer sequences derived by the method of addition modulo a prime number with different addends m and n, $A = \{a_0, \ldots, a_{N-1}\}$ and $B = \{b_0, \ldots, b_{N-1}\}$, there must exist an integer $t \in [0, \ldots, N-1]$ such that $a_t = b_t$.

Proof: Presented in the full version of the paper [18]. ■

Theorem 3: For two nodes that select two different CH sequence constructed by Algorithm 4, there must be at least N overlapping slots within a hopping period between the two CH sequences no matter how their hopping processes are misaligned.

Proof: Presented in the full version of the paper [18]. ■

VIII. Discussion

Complexity of the CH Sequence Construction Algorithms: In a CR network with N rendezvous channels, we compare the complexity of ETCH's CH construction algorithms to those of existing solutions as follows. For SYNC-ETCH, both the two-phase algorithm (i.e., Algorithm 1) and the single-phase algorithm (i.e., [18, Algorithm 5]) have a complexity of $O(N^2)$. In L-QCH and M-QCH [5], the CH sequence construction algorithm complexity is $O(|S| \cdot N^2)$, where $|S|$ represents the size of the quorum system being used. Since $|S|$ and N have the same order of magnitude, the algorithm complexity of M-QCH and L-QCH is $O(N^3)$. For ASYNC-ETCH, the CH sequence construction algorithm (i.e., Algorithm 4) has a complexity of $O(N^3)$. A-QCH [5] has a complexity of $O(|S| \cdot N) = O(N^2)$, and SeqR's complexity is also $O(N^2)$ [16]. From the comparison, we can see that SYNC-ETCH has the same CH construction algorithm complexity as the existing solutions. For ASYNC-ETCH, the algorithm complexity is an order of magnitude higher than the existing solutions. However, in exchange, ASYNC-ETCH achieves better channel load balance and TTR once the CH sequences are constructed.

ASYN-ETCH: When N Is Not Prime: Our previous analysis of SYNC-ETCH assumes that N (i.e., the number of rendezvous channels) is a prime number. To address the practical issue when N is not a prime number, we can make the following adjustment: ASYNC-ETCH picks the smallest prime number that is greater than the number of rendezvous channels as the parameter N for Algorithm 4 and maps the excessive rendezvous channels down to the actual rendezvous channels correspondingly. Fig. 8 demonstrates an example of ASYNC-ETCH CH sequences construction in a CR network with $N = 4$ rendezvous channels C_0–C_3. ASYNC-ETCH first constructs four integer sequences A_0–A_3 using addition modulo a prime number 5 with addends 1 to 4, respectively. Then, it converts the integer sequences A_i to A'_i $(0 \leq i \leq 3)$ by replacing all those numbers $m (m \geq N, N = 4$ in our

TABLE II
COMPARISONS BETWEEN COMMUNICATION RENDEZVOUS PROTOCOLS

	Avg. Rend. channel load	Average TTR	Rend. channels utilization ratio
M-QCH	$\frac{2}{3}$	$\frac{3}{2}$	$\frac{1}{N}$
L-QCH	$\approx \frac{1}{\sqrt{2N-1}}$	$\frac{2N-1}{2}$	$\frac{1}{N}$
SYNC-ETCH	$\frac{1}{N}$	$\frac{2N-1}{2}$	1
A-QCH	$\frac{1}{2}$	$\geq \frac{9}{2}$	N/A
SeqR	$\frac{1}{N}$	$\frac{N^2+N}{2}$	N/A
ASYNC-ETCH	$\frac{1}{N}$	$\frac{2N^2+N}{N-1} \approx 2N$	N/A

example) in A_i $(0 \leq i \leq 3)$ with number m mod N (i.e., the remainder of m divided by N). Thus, in our example, the number 4 in A_0 to A_3 is replaced by the number 0 in A'_0 to A'_3. Then, the ASYNC-ETCH CH subsequences will be constructed according to integer sequences of A'_0 to A'_3. The drawback of this method is that some rendezvous channels are assigned more times to the CH sequences. Therefore, for CR networks using ASYNC-ETCH, we recommend to assign a prime number of channels for control information exchange.

Network Models: Our discussion so far is based on a network model where communications occur directly between two CR ndoes. Practical applications of such single-hop network model include single-hop point-to-point communication and infrastructure networks where multiple endpoints communicate via an access point. The proposed scheme can also be used with a multiple-hop network model. In the multiple-hop scenario, ETCH will be applied in a hop-by-hop manner, where different pairs of nodes rendezvous using ETCH independently.

IX. COMPARISONS

In this section, we compare the theoretical performance of ETCH to that of QCH [5] and SeqR [16]. The comparisons are divided into two groups. We compare SYNC-ETCH to M-QCH and L-QCH, all of which hold the global clock synchronization assumption, in the first group, and compare the three asynchronous protocols, ASYNC-ETCH, A-QCH, and SeqR, in the second group, using the three metrics introduced in Section III: average rendezvous channel load, average TTR, and rendezvous channels utilization ratio. Note that choosing either the two-phase algorithm or the single-phase algorithm for CH sequence construction in SYNC-ETCH makes no difference on the protocol's theoretical performances because in these theoretical comparisons, we do not consider the impacts of the appearances of primary users, which we will evaluate later in Section X-B.

Table II summarizes the comparison results, where N is the number of rendezvous channels of the CR network. In the synchronous protocols group, we pick parameters for L-QCH such that it produces the same number of CH sequences as SYNC-ETCH for the purpose of fair comparison. SYNC-ETCH outperforms M-QCH and L-QCH on the metrics of average rendezvous channel load and rendezvous channels utilization ratio because in every hopping slot it efficiently utilizes all rendezvous channels in establishing control channels, while there is only one channel can be used as control channel in each hopping slot with M-QCH and L-QCH. Thus, theoretically, SYNC-ETCH experiences less traffic collisions and achieves higher throughput than QCH. For the metric of average TTR, M-QCH achieves the best theoretical performance. However, it has a very large average load on each rendezvous channel ($\frac{2}{3}$ of all the network nodes use the same rendezvous channel), which will cause a high probability of traffic collisions and further make the time-to-rendezvous performance of M-QCH worse than its theoretical value in practice.

In the asynchronous protocols group, A-QCH has the worst performance in terms of average rendezvous channel load because it only ensures two of the rendezvous channels can be used as control channels, while both ASYNC-ETCH and SeqR utilize all the rendezvous channels in control channel establishment. Moreover, A-QCH cannot provide a bounded TTR. SeqR, which constructs only one CH sequence, can only guarantee one overlapping slot in a hopping period. Thus, the average TTR for SeqR is half of the number of slots in the CH sequence (i.e., $\frac{N^2+N}{2}$). For ASYNC-ETCH's performance on the metric of average TTR, we make the following analysis: We proved in Section VII-C that for the cases that when two nodes select the same CH sequence and when they select two different CH sequences, they are respectively guaranteed to meet in at least one slot and at least N slots within a hopping period. Since ASYNC-ETCH generates $N-1$ different CH sequences and the CH sequence selection is random, on average there are $\frac{1}{N-1} + \frac{(N-2)N}{N-1} = N-1$ guaranteed overlapping slots in a hopping period. Hence, the average TTR for ASYNC-ETCH is $\frac{2N^2+N}{N-1} \approx 2N$.

X. PERFORMANCE EVALUATION

A. Comparing ETCH to the Existing CH-Based Communication Rendezvous Protocols

1) Methodology: We evaluate ETCH by comparing it to QCH and SeqR in the `ns` -2 simulator. We divide the evaluation into two portions based on the assumption about the existence of global clock synchronization. In Section X-A.2, we compare the performances of SYNC-ETCH (using the two-phase algorithm for CH sequence construction), M-QCH and L-QCH. In Section X-A.3, we compare the performances of ASYNC-ETCH with A-QCH and SeqR.

In the evaluation, we modify the `ns` -2 simulator to make it be able to perform multichannel wireless communication simulations based on the Hyacinth project [21]. In our simulations, there are a varying number of nodes in a 500×500-m^2 area, where each of the nodes is in all other nodes' communication ranges. The length of a hopping slot is set to 100 ms. We establish constant bit rate (CBR) flows, where the packet size is set to 800 B and the packet rate is 125 packets/s, from each node to all other nodes. These flows are started and stopped randomly during the simulation such that there is no more than one flow from the same node is activated simultaneously (because there is only one transceiver equipped with each node). Hyacinth's manual routing protocol is used in routing packets between the nodes. We disable the RTS/CTS function in the simulator and rely on the retransmission mechanism to deal with packet collisions. In the simulations, the CR network has five rendezvous channels (i.e., $N = 5$), each of which can possibly be used by the primary user. To simplify the simulation, we suppose all the secondary users are within the communication range of the primary user. The appearances of the primary user is simulated as

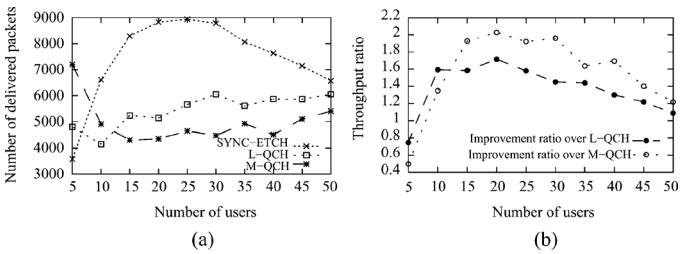

Fig. 9. Throughput performances of the synchronous protocols. (a) Traffic throughput. (b) SYNC-ETCH's throughput improvement ratio.

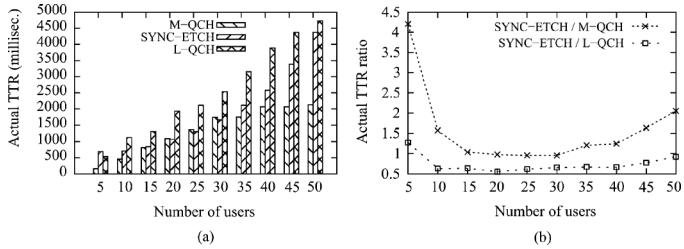

Fig. 10. TTR performances of the synchronous protocols. (a) Actual TTR. (b) Actual TTR ratio.

follows. We first decide whether the primary user shows or not by flipping a coin. If the primary user appears, we randomly disable a rendezvous channel for another random period of time. Otherwise, all the rendezvous channels are made to be available to the nodes also for a random period of time before we flip the coin again (the random periods above are set between one slot and the number of slots of 10 periods). We repeat this process during the entire simulation.

2) Synchronous Communication Rendezvous Protocols: We conduct two simulation experiments to study the performances of the synchronous protocols on traffic throughput and actual TTR. In each experiment, we run the simulation for 10 rounds with different numbers of secondary users (from 5 to 50 with a step length of 5) in each round.

Fig. 9 shows the traffic throughput performances of the three synchronous protocols. Fig. 9(a) shows the actual throughput, while (b) illustrates the performance improvement ratio curves of SYNC-ETCH over L-QCH and M-QCH. SYNC-ETCH has a lower throughput than L-QCH and M-QCH when there are five secondary users in the network. This is because in CH sequences of L-QCH and M-QCH, rendezvous channels are randomly assigned to those non-frame-channel-slots, which may give a pair of nodes using L-QCH or M-QCH extra slots to rendezvous in other than the frame-channel-slot. This is also because there are no or little collisions when there are only five nodes. However, when the number of secondary users is equal or greater than 10, SYNC-ETCH achieves higher traffic throughput than L-QCH and M-QCH, especially when the nodes-channels ratio is in the range of 3–6 (i.e., when there are 15–30 nodes). In this case, traffic collision dominates the factors that influence the throughput performance. With both L-QCH and M-QCH, nodes are always compete for one rendezvous channel as control channel leaving all other rendezvous channels unused in a hopping frame, which causes a high probability of collisions when the nodes–channels ratio is bigger than 1. On the contrary, SYNC-ETCH schedules rendezvous among its CH sequences such that all the rendezvous channels can be utilized in every hopping slot. This approach greatly reduces traffic collisions and hence increases throughput. Furthermore, it can be also noticed in Fig. 9 that the throughput performance of the three synchronous protocols converges as the nodes–channels ratio approaches 10. This is because collisions dominate traffics in each rendezvous channel with all the synchronous protocols. In this case, we suggest to assign more rendezvous channels to accommodate such a high number of secondary users based on the expected network coverage [22] and channel quality [23].

Fig. 10(a) shows the TTR performances of the three synchronous protocols, and (b) demonstrates the TTR ratios of SYNC-ETCH over L-QCH and M-QCH. The TTRs of the three protocols increase as the number of secondary users grows because of the increasing traffic collisions. Although M-QCH achieves the best TTR performance among the three as analyzed in Section IX, it does not get the theoretical TTR performance boost over SYNC-ETCH as analyzed in Section IX. Theoretically, M-QCH performs 3 times better than SYNC-ETCH in TTR, because it has an average TTR of 1.5 while SYNC-ETCH's value is 4.5. However, the simulation results show that M-QCH cannot achieve the theoretical performance advantage over SYNC-ETCH: M-QCH is only 1.5 times better (i.e., on average, SYNC-ETCH's actual TTR is only 1.5 times of that of M-QCH). The reason of M-QCH's TTR performance degradation in the simulation experiment is because the nodes using M-QCH experience more severe traffic collisions than those using SYNC-ETCH, and therefore cannot achieve the theoretical performance advantage.

From the above two simulations, it can be seen that SYNC-ETCH achieves the best balance between traffic throughput and TTR among the three synchronous protocols.

3) Asynchronous Communication Rendezvous Protocols: In this section, we compare the throughput and the TTR performances between the three asynchronous protocols: ASYNC-ETCH, A-QCH, and SeqR.

Fig. 11 shows the performances of the three asynchronous protocols. In Fig. 11(a), the traffic throughput performances are shown. ASYNC-ETCH performs constantly better than the other two protocols in this metric. This is because ASYNC-ETCH is able to utilize all the rendezvous channels as control channels while A-QCH uses only two of them. Meanwhile, ASYNC-ETCH improves on SeqR such that it achieves a shorter average TTR, which contributes to the throughput performance boost over SeqR. Fig. 11(b) shows the actual TTR performances of the three protocols. It is not surprising that ASYNC-ETCH performance better than SeqR because ASYNC-ETCH's average TTR is shorter than that of SeqR (see Table II for details). For A-QCH, we construct CH sequences such that they have an average TTR of 4.5, which is the best that A-QCH is able to achieve. Even so, ASYNC-ETCH still performs better than A-QCH.

B. Comparing the Two Algorithms for CH Sequence Construction in SYNC-ETCH

To quantize how even the N rendezvous channels (i.e., $C_0, \ldots, C_{N-1}$) appear in a CH sequence S, we define the

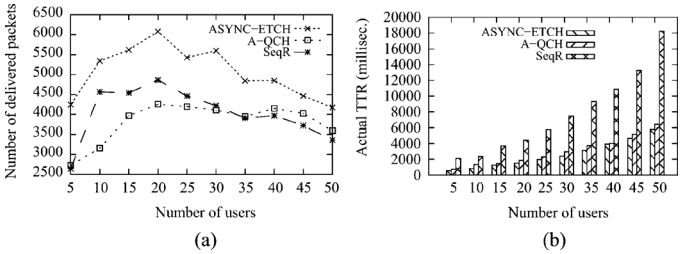

Fig. 11. Throughput and TTR of the asynchronous protocols. (a) Traffic throughput. (b) Actual TTR.

"evenness score" of S regarding rendezvous channel appearance probability as

$$\varepsilon_S = \sqrt{\frac{\sum_{i=0}^{N-1}(a_i - \frac{|S|}{N})^2}{N}}$$

where $|S|$ is the number of hopping slots of S, and a_i is the number of hopping slots of S in which channel C_i appears. We further convert ε_S into a normalized score $N(\varepsilon_S)$, which is the range of $[0, 1]$ and can be expressed as

$$N(\varepsilon_S) = 1 - \frac{\varepsilon_S - \varepsilon_{\text{best}}}{\varepsilon_{\text{worst}} - \varepsilon_{\text{best}}}.$$

In $N(\varepsilon_S)$, $\varepsilon_{\text{best}}$ and $\varepsilon_{\text{worst}}$ are the evenness scores of the best case and the worst case of fulfilling the even use of rendezvous channels requirement, respectively. In the best case, each of the N rendezvous channels appears in S with the same the same probability, while in the worst case, a single channel appears in all the hopping slots of S. For instance, with the SYNC-ETCH protocol where there are $2N-1$ hopping slots in a CH sequence, the best case that a CH sequence S satisfies the even use of rendezvous channels requirement is that a rendezvous channels appears once in S, while each of the remaining $N-1$ channels appears twice in S. The evenness score of the best case is calculated as $\varepsilon_{\text{best}} = \sqrt{\frac{(1-\frac{2N-1}{N})^2+(N-1)(2-\frac{2N-1}{N})^2}{N}}$. In the worst case, all the $2N-1$ slots are assigned with the same CH sequence. Accordingly, the evenness score of the worst case is calculated as $\varepsilon_{\text{worst}} = \sqrt{\frac{((2N-1)-\frac{2N-1}{N})^2+(N-1)(0-\frac{2N-1}{N})^2}{N}}$.

Low normalized evenness score of a CH sequence S indicates that S uses one or several rendezvous channels more than the remaining channels, which causes the nodes selecting S to have higher probability to experience communication outages if the primary users of those heavily relied channels show up.

In SYNC-ETCH, every CH sequence constructed by the single-phase algorithm has a normalized evenness score of 1, which is the optimal case of fulfilling the even use of rendezvous channels requirement. To evaluate how well the two-phase algorithm satisfies this requirement, we calculate the average value and the corresponding standard deviation of the evenness scores of the $2N$ CH sequences constructed by the two-phase algorithm. Fig. 12 shows the results of the cases where the value of N ranges from 3 to 99. The top graph of Fig. 12 plots the average value of the evenness scores, and the bottom graph plots the corresponding standard deviations. We can see from the results that the two-phase algorithm still achieves an average normalized evenness score that is larger than 0.9 when N is greater than 10, and that the averaged score increases as N increases.

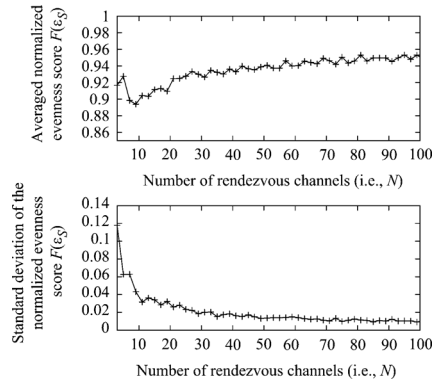

Fig. 12. Channel appearance evenness score of the two-phase CH sequence construction algorithm. (The evenness score of the single-phase algorithm is always 1.)

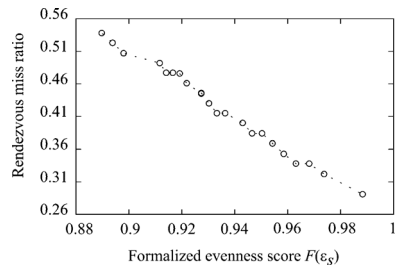

Fig. 13. Rendezvous miss ratio versus channel appearance evenness score.

We further perform an experiment to evaluate how the normalized evenness scores of CH sequences affect the performances of the communication rendezvous protocol. In the experiment, we let a node A that is stuck to a fixed CH sequence S_i rendezvous with another node B for $2N-1$ times, where the node B selects a different CH sequence S_j ($j \neq i$) at each time. We disable γ ($0 < \gamma < 1$) of the rendezvous channels that are used most frequently in S_i. The node A fails to rendezvous with the node B at a time if the overlapping channel between S_i and S_j is disabled. We then calculate "rendezvous miss ratio" of the CH sequence S_i as the ratio between the number of times when a rendezvous attempt fails and the total number of rendezvous attempts (i.e., $2N-1$). Fig. 13 plots the relationship between the normalized evenness score and the rendezvous miss ratio of a CH sequence constructed by the two-phase algorithm S when $N = 33$ and $\gamma = 0.3$. Under the same settings, the rendezvous miss ratio of a CH sequence constructed by the single-phase algorithm is 0.27.

XI. Conclusion

We have presented ETCH, efficient channel-hopping-based communication rendezvous protocols for CR networks. ETCH protocols include SYNC-ETCH and ASYNC-ETCH. SYNC-ETCH, which assumes global clock synchronization, efficiently utilizes all the rendezvous channels in establishing control channels all the time. ASYNC-ETCH is able to make a pair of nodes rendezvous without being synchronized. Using a combination of theoretical analysis and simulations, we show that ETCH protocols perform better than the existing solutions for communication rendezvous in CR networks.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their insightful comments. This paper is based on the conference paper [1]. The major improvement of this paper is that a single-phase CH sequence construction algorithm (Section VI) for SYNC-ETCH is proposed, which can *guarantee* all the rendezvous channels are evenly used.

REFERENCES

[1] Y. Zhang, Q. Li, G. Yu, and B. Wang, "ETCH: Efficient channel hopping for communication rendezvous in dynamic spectrum access networks," in *Proc. IEEE INFOCOM*, 2011, pp. 2471–2479.

[2] M. Song, C. Xin, Y. Zhao, and X. Cheng, "Dynamic spectrum access: From cognitive radio to network radio," *IEEE Wireless Commun.*, vol. 18, no. 1, pp. 23–29, Feb. 2012.

[3] Y. Zhao, M. Song, C. Xin, and M. Wadhwa, "Spectrum sensing based on three-state model to accomplish all-level fairness for co-existing multiple cognitive radio networks," in *Proc. IEEE INFOCOM*, 2012, pp. 1782–1790.

[4] J. Zhao, H. Zheng, and G. H. Yang, "Distributed coordination in dynamic spectrum allocation networks," in *Proc. IEEE DySPAN*, Dec. 2005, pp. 259–268.

[5] K. Bian, J. Park, and R. Chen, "A quorum-based framework for establishing control channels in dynamic spectrum access networks," in *Proc. ACM MobiCom*, Jun. 2009, pp. 25–36.

[6] L. Ma, X. Han, and C. Shen, "Dynamic open spectrum sharing MAC protocol for wireless ad hoc networks," in *Proc. IEEE DySPAN*, 2005, pp. 203–213.

[7] B. Hamdaoui and K. G. Shin, "OS-MAC: An efficient MAC protocol for spectrum-agile wireless networks," *IEEE Trans. Mobile Comput.*, vol. 7, no. 8, pp. 915–930, Aug. 2008.

[8] H. Su and X. Zhang, "Cross-layer based opportunistic MAC protocols for QoS provisionings over cognitive radio wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 1, pp. 118–129, Jan. 2008.

[9] Z. Qin, Q. Li, and G. Hsieh, "Defending against cooperative attacks in cooperative spectrum sensing," *IEEE Trans. Wireless Commun.*, vol. 12, no. 6, pp. 2680–2687, Jun. 2013.

[10] T. Chen, H. Zhang, G. M. Maggio, and I. Chlamtac, "CogMesh: A cluster-based cognitive radio network," in *Proc. IEEE DySPAN*, 2007, pp. 168–178.

[11] T. Chen, H. Zhang, M. D. Katz, and Z. Zhou, "Swarm intelligence based dynamic control channel assignment in CogMesh," in *Proc. IEEE ICC*, 2008, pp. 123–128.

[12] L. Lazos, S. Liu, and M. Krunz, "Spectrum opportunity-based control channel assignment in cognitive radio networks," in *Proc. IEEE SECON*, 2009, pp. 1–9.

[13] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad hoc wireless networks," in *Proc. ACM MobiCom*, Sep. 2004, pp. 216–230.

[14] *Bluetooth Special Interest Group Specification of the Bluetooth System*, V1.1.

[15] A. Tzamaloukas and J. J. Garcia-Luna-Aceves, "Channel-hopping multiple access," in *Proc. IEEE ICC*, 2000, pp. 415–419.

[16] L. A. DaSilva and I. Guerreiro, "Sequence-based rendezvous for dynamic spectrum access," in *Proc. IEEE DySPAN*, Oct. 2008, pp. 1–7.

[17] Z. Lin, H. Liu, X. Chu, and Y.-W. Leung, "Jump-stay based channel-hopping algorithm with guaranteed rendezvous for cognitive radio networks," in *Proc. IEEE INFOCOM*, 2011, pp. 2444–2452.

[18] Y. Zhang, G. Yu, Q. Li, H. Wang, X. Zhu, and B. Wang, "Toward efficient channel hopping for communication rendezvous in dynamic spectrum access networks," WM CS Tech. Rep., WM-CS-2012–03, 2012.

[19] "Matching (graph theory)," 2013 [Online]. Available: http://en.wikipedia.org/wiki/Matching_(graph_theory)

[20] T. D. LeSaulnier, C. Stocker, P. S. Wenger, and D. B. West, "Rainbow matching in edge-colored graphs," *Electron. J. Combin.*, vol. 17, no. 26, 2010.

[21] T. Chiueh, A. Raniwala, R. Krishnan, and K. Gopalan, "Hyacinth: An IEEE 802.11-based multi-channel wireless mesh network," 2005 [Online]. Available: http://www.ecsl.cs.sunysb.edu/multichannel/

[22] Z. Yu, J. Teng, X. Li, and D. Xuan, "On wireless network coverage in bounded areas," in *Proc. IEEE INFOCOM*, 2013, to be published.

[23] X. Xing, T. Jing, Y. Huo, H. Li, and X. Cheng, "Channel quality prediction based on Bayesian inference in cognitive radio networks," in *Proc. IEEE INFOCOM*, 2013, to be published.

Yifan Zhang received the B.S. degree in computer science from the Beihang University, Beijing, China, in 2004, and is currently pursuing the Ph.D. degree in computer science at the College of William and Mary, Williamsburg, VA, USA.

His research interests include wireless networks, mobile computing systems, and operating systems.

Gexin Yu graduated from the University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 2006.

He is an Associate Professor with the College of William and Mary, Williamsburg, VA, USA. He was a non-tenure-track Assistant Professor with Vanderbilt University, Nashville, TN, USA, from 2006 to 2008. His primary research is on graph theory and its applications.

Qun Li (M'05–SM'12) received the Ph.D. degree in computer science from Dartmouth College, Hanover, NH, USA, in 2004.

He is an Associate Professor with the Department of Computer Science, College of William and Mary, Williamsburg, VA, USA. His research interests include wireless networks, sensor networks, RFID, and pervasive computing systems.

Dr Li received the NSF Career Award in 2008.

Haodong Wang (A'09) received the Ph.D. degree in computer science from the College of William and Mary, Williamsburg, VA, USA, in 2009.

He is an Assistant Professor with the Department of Computer and Information Science, Cleveland State University, Cleveland, OH, USA. His research interests include wireless networks, sensor networks, security and privacy, pervasive computing systems, and software defined radio.

Xiaojun Zhu (S'13) received the B.S. degree in computer science from Nanjing University, Nanjing, China, in 2008, and is currently pursuing the Ph.D. degree in computer science at the same university.

His research interests include RFID systems, smartphone systems, wireless sensor networks, and vehicular networks.

Baosheng Wang received the Ph.D. degree in computer science from the National University of Defense Technology (NUDT), Changsha, China, in 2005.

He is currently a Professor with the Computer School, NUDT. His research interests include computer networks, wireless network, and cluster computing.