An Enhanced Fast Multi-Radio Rendezvous Algorithm in Heterogeneous Cognitive Radio Networks

Yeh-Cheng Chang, Cheng-Shang Chang and Jang-Ping Sheu
Department of Computer Science and Institute of Communications Engineering
National Tsing Hua University
Hsinchu 30013, Taiwan, R.O.C.
Email: jas1123kimo@gmail.com; cschang@ee.nthu.edu.tw; sheujp@cs.nthu.edu.tw

***Abstract*—In this paper, we propose a fast rendezvous algorithm for a heterogeneous cognitive radio network (CRN), where each user might have more than one radio. One of the well-known problems for most multi-radio rendezvous algorithms in the literature is that they are not backward compatible to users with only *one* radio. To tackle this backward compatibility problem, our approach is a hierarchical construction that groups several time slots into an interval and proposes a novel algorithm to emulate two radios with a single radio in an interval. By doing so, at the interval level, each user behaves as if it had (at least) two radios. For the two-user rendezvous problem in a CRN with N commonly labelled channels, the interval length is chosen to be M time slots, where $M = 2\lceil \log_2(\lceil \log_2 N \rceil)\rceil + 7$. We show that the maximum time-to-rendezvous (MTTR) of our algorithm is bounded above by $9M\lceil n_1/m_1 \rceil \cdot \lceil n_2/m_2 \rceil$ time slots, where n_1 (resp. n_2) is the number of available channels to user 1 (resp. 2), and m_1 (resp. m_2) is the number of radios for user 1 (resp. 2). For the setting that each user is equipped with only one radio and two available channels, our MTTR bound is only M and that improves the state-of-the-art bound $16(\lceil \log_2 \log_2 N \rceil + 1)$ in the literature. By conducting extensive simulations, we show that for the expected time-to-rendezvous (ETTR), our algorithm is also better than several existing multi-radio algorithms.**

keywords: multichannel rendezvous, maximum time-to-rendezvous, multiple radios

I. The multichannel rendezvous problem

Rendezvous search that asks two persons to find each other among a set of possible locations is perhaps one of the most common problems in our daily life. Such a problem has been studied extensively in the literature (see e.g., the book [2] and references therein). Motivated by the problem of establishing a control channel between two secondary spectrum users in a cognitive radio network (CRN), the rendezvous search problem has regained tremendous research interest lately. In a CRN, there are two types of users: primary spectrum users (PUs) and secondary spectrum users (SUs). PUs usually have the licence to use the spectrum assigned to them. On the other hand, SUs are only allowed to share spectrum with PUs provided that they do not cause any severe interference to PUs. To do this, SUs first sense a number of frequency channels. If a channel is not blocked by a PU, then that channel is available to that SU and it may be used for establishing a communication link. One of the fundamental problems in a CRN is then for two SUs to find a common available channel and such a problem is known as the multichannel rendezvous problem.

The conference version of this paper [1] was accepted for presentation in ICC 2018.

The objective of this paper is to provide a fast rendezvous algorithm for the multichannel rendezvous problem with multiple radios. In the traditional setting of the multichannel rendezvous problem, the number of radio for a user is assumed to be 1. It is a common wisdom that increasing the number of radios for each user can speed up the rendezvous process. For this, let us consider a CRN with N channels (with $N \geq 2$), indexed from 0 to $N-1$. Time is slotted (the discrete-time setting) and indexed from $t = 0, 1, 2, \ldots$. There are two users who would like to rendezvous on a common available channel by hopping over these N channels with respect to time. The available channel set for user i, $i = 1, 2$, is

$$\mathbf{c}_i = \{c_i(0), c_i(1), \ldots, c_i(n_i - 1)\},$$

where $n_i = |\mathbf{c}_i|$ is the number of available channels to user i, $i = 1, 2$. We assume that there is at least one channel that is commonly available to the two users, i.e.,

$$\mathbf{c}_1 \cap \mathbf{c}_2 \neq \phi. \tag{1}$$

Moreover, we assume that user i has m_i radios, where $m_i \geq 1$, $i = 1$ and 2. Denote by $X_1(t)$ (resp. $X_2(t)$) the set of channels selected by user 1 (resp. user 2) on its m_i radios at time t. Then the time-to-rendezvous (TTR), denoted by T, is the number of time slots (steps) needed for these two users to select a common available channel, i.e.,

$$T = \inf\{t \geq 0 : X_1(t) \cap X_2(t) \neq \phi\} + 1, \tag{2}$$

where we add 1 in (2) as we start from $t = 0$.

In this paper, we do not assume that the clocks of these two users are synchronized. In order to guarantee rendezvous within a finite number of time slots, we will construct periodic Channel Hopping (CH) sequences based on the available channel set to a user. For the setting that each user has two radios, the rendezvous search is rather simple as shown in [3]. In that setting, user i can select two different primes $p_{i,0}$ and $p_{i,1}$ (with $p_{i,0} < p_{i,1}$) not smaller than n_i, and run the modular clock algorithm in [4] with $p_{i,0}$ (resp. $p_{i,1}$) for the first (resp. second) radio of user i, $i = 1$ and 2. Since there is at least one prime selected by user 1 that is different from one of the two primes selected by user 2, it follows from the Chinese Remainder Theorem that these two users will rendezvous within $p_{1,1} \cdot p_{2,1}$ time slots. Let us simply call such an algorithm the *two-prime modular clock algorithm*.

But the problem arises when each user only has a single radio in the traditional setting. Then the Chinese Remainder Theorem may not be applicable as they may select the same prime. Such a problem also arises in many multi-radio schemes, e.g. GCR [3], RPS [5], AMRR [6], proposed in the literature. To address the backward compatibility problem that some users in a CRN might only have a single radio, our idea is to emulate each radio of a user as it were two radios. Specifically, we will construct CH sequences that group several slots into an *interval* and within an interval each radio selects two channels according to the two-prime modular clock algorithm. When a user has more than one radio, we simply divide its available channels as evenly as possible to its radio(s). By doing so, the number of channels assigned to each radio is much smaller and thus the primes selected by each radio are also smaller. As a result, the maximum TTR (MTTR) can be greatly reduced.

We summarize our contributions as follows:

(i) For the two-user rendezvous problem in a CRN with N commonly labelled channels, we propose a fast rendezvous algorithm with the MTTR bounded above by $9M\lceil n_1/m_1\rceil \cdot \lceil n_2/m_2\rceil$ time slots, where $M = 2\lceil\log_2(\lceil\log_2 N\rceil)\rceil + 7$.

(ii) Our algorithm is backward compatible with users equipped with a single radio. For the setting that each user is equipped with only one radio and two available channels, our MTTR bound is only M and that improves the state-of-the-art bound $16(\lceil\log_2\log_2 N\rceil + 1)$ in [7].

(iii) By conducting extensive simulations, we show that for the expected time-to-rendezvous (ETTR), our algorithm outperforms several existing algorithms, including JS/I and JS/P [5], GCR [3], RPS [5], and AMRR [6]. Furthermore, even for the setting where each user is equipped with more than one radio, the (measured) MTTR of our algorithm is comparable to those in these algorithms (that do not have bounded MTTR when a user is equipped with a single radio).

The rest of this paper is organized as follows: In Section II, we discuss related works on the multichannel rendezvous problem. We then introduce the generic multi-radio CH sequences in Section III. Further improvements of our generic CH sequences are shown in Section IV. To show the effectiveness of our algorithm, we conduct extensive simulations to compare the performance of our algorithm with several existing algorithms in Section V. The paper is then concluded in Section VI.

II. RELATED WORKS

In this section, we first introduce the taxonomy of the multichannel rendezvous problem. For a more detailed introduction of the multichannel rendezvous problem, we refer readers to the tutorial [8] and the book [9]. We then discuss some recent works on multi-radio CH schemes.

The channel hopping schemes for the multichannel rendezvous problem are in general classified as follows:

Symmetric/Asymmetric: The first classification is based on whether users can be assigned different roles in the rendezvous process. In asymmetric algorithms (see e.g., [10] , [11], [12], [13]), a user is assigned the role of a sender or receiver. Users assigned with different roles then follow different strategies to generate their CH sequences. For instance, the receiver can stay at the same channel while the sender cycles through all the available channels. Since users follow different strategies, the time-to-rendezvous can be greatly reduced in the asymmetric setting. On the other hand, in a symmetric CH scheme, users follow the same strategy to generate their CH sequences and it is much more difficult to guarantee rendezvous.

Onymous/Anonymous: Onymous algorithms (see e.g., [11], [14], [15], [16], [17], [19], [20]) assume that each user is assigned with a unique identifier (ID), e.g., a MAC address. As such, users can use their unique IDs to play different roles to speed up the rendezvous process. The hard part of this approach is to find an efficient mapping of IDs that minimizes the MTTR.

Synchronous/Asynchronous: Synchronous algorithms (see e.g., [21], [22]) assume that the clocks of users are synchronized so that they have a common index of time. As such, users can start their CH sequences *synchronously* to speed up the rendezvous process. For instance, users can have the same jump pattern and rendezvous in every time slot [23]. Clock synchronization is easy to obtain if there is a global clock source, e.g., GPS or base stations. However, in a distributed environment it might not be practical to assume that the clocks of two users are synchronized as they have not rendezvoused yet. Without clock synchronization, guaranteed rendezvous is much more difficult. In the literature, there are various asynchronous algorithms that have bounded MTTR (see e.g., [7], [11], [12], [13], [14], [15], [16], [17], [24], [25], [26], [27]).

Homogeneous/Heterogeneous: In a homogenous environment, the available channel sets of the two users are assumed to be the same. On the other hand, the available channel sets of the two users might be different in a heterogeneous environment. Two users that are close to each other are likely to have the same available channel sets. Due to the limitation of the coverage area of a user, two users tend to have different available channel sets if they are far apart. Rendezvous in a homogeneous environment is in general much easier than that in a heterogeneous environment. There are various heterogeneous CH algorithms that have bounded MTTR (see e.g., [7], [11], [12], [24], [25], [27]). We note that in the literature some authors refer a homogenous (resp. heterogeneous) environment as a symmetric (resp. asymmetric) environment.

Oblivious/Non-oblivious: In most previous works for the multichannel rendezvous problem, it is commonly assumed that there is a universal channel labelling. As such, it is possible for a user to learn from a failed attempt to rendezvous. In particular, a universal channel labelling allows channels to be ordered and that piece of information can be used for speeding up the rendezvous process (see e.g., [7], [25]). On the other hand, oblivious rendezvous (see e.g., [3], [6], [15],

[19]) is referred to the setting where nothing can be learned from a failed attempt to rendezvous. In such a setting, the available channel sets are different (heterogeneous) and there is no role assignment (symmetric), no clock synchronization (asynchronous), and no universal channel labelling. Such a setting is the most challenging setting of the multichannel rendezvous problem.

Single-radio/Multi-radio: Recently, several research works focus on the multi-radio CH schemes [3], [5], [6]. Here, users are assumed to be equipped with multiple radios. As a result, they can generate CH sequences that hop on more than one channel in a time slot. This improves the probability of rendezvous and thus shortens the time-to-rendezvous.

In this paper, we focus on the symmetric, anonymous, asynchronous, and heterogeneous model with multi-radios. As discussed in Section I, we do assume that there is a universal channel labelling. In the following, we briefly review several multi-radio CH schemes in such an environment. Yu et al. [5] proposed various CH algorithms for multiple radios that utilize the CH sequences from a single radio setting, e.g., the Jump-Stay (JS) sequence [24]. The JS/I strategy in [5] simply assigns every radio the JS sequence that starts *independently* from a time slot in the period of the JS sequence. On the other hand, the JS/P strategy [5] assigns the JS sequence in the round-robin fashion to the radios of each user. As such, the JS/P strategy does not guarantee rendezvous if the two users do not have the same number of radios. They also proposed a CH algorithm, called the role-based parallel sequence (RPS). In general, RPS partitions the radios of a user into two modes: the stay mode and the jump mode. It reserves one radio in the stay mode and that radio stays on one particular channel for a specific period, while the other radios in the jump mode hop among the channels. It was shown that RPS can guarantee rendezvous within $O(P(N-G)/(\min(m_1,m_2)-1)$ time slots, where N is the number of channels, P is a prime not less than N, G is the number of common channels, and m_1 (resp. m_2) is the number of radios for user 1 (resp. 2).

In [6], Yu et al. proposed the adjustable version of RPS, called Adjustable Multi-radio Rendezvous (AMRR). Instead of using a single radio as the stay radio in [5], AMRR provides a control knob for the users to optimize MTTR or ETTR by making the number of stay radios adjustable. In general, the MTTR of AMRR can be improved by increasing the number of stay radios and the ETTR can be improved by decreasing the number of stay radios. Also, AMRR can be used in heterogeneous CRNs and this is more general than the homogeneous assumption in [5]. The MTTR of AMRR was shown to be $O((n_1 n_2)/(m_1 m_2))$, where n_1 and n_2 are the numbers of available channels for user 1 and user 2, respectively. The bound is independent of the total number of channels N. But it only holds when $m_1 > 1$ and $m_2 > 1$.

Li et al. proposed a General Construction for Rendezvous (GCR for short) in [3]. As described in Section I, GCR utilizes the two-prime modular clock algorithm to guarantee rendezvous. For this, they first arrange the available radios into pairs and divide the available channels into these pairs of radios. Due to this pairing strategy, GCR requires the number of radios to be an even number (for a better MTTR bound). It was shown in [3] that the MTTR of GCR is $O((n_1 n_2)/(m_1 m_2))$. As in AMRR, this bound only holds when $m_1 > 1$ and $m_2 > 1$.

We note that RPS [5], AMRR [6] and GCR [3] cannot guarantee a bounded MTTR in a single radio environment. A comparison table for the MTTRs of these multi-radio algorithms is given in Table I.

TABLE I
COMPARISONS OF THE MTTRS OF VARIOUS MULTI-RADIO RENDEZVOUS ALGORITHMS

Algorithm	**MTTR**	**Applicability in single radio**
RPS [5]	$O(\frac{P(N-G)}{\min(m_1,m_2)-1})$	no
JS/I [5]	$O(NP(P-G))$	yes
JS/P [5]	$O(\frac{NP(P-G)}{m})$	yes
AMRR [6]	$O(\frac{n_1 n_2}{m_1 m_2})$	no
GCR [3]	$O(\frac{n_1 n_2}{m_1 m_2})$	no
OURS	$9M\lceil n_1/m_1\rceil \cdot \lceil n_2/m_2\rceil$	yes

N is the total number of channels, P is a prime not less than N, $M = 2\lceil \log_2(\lceil \log_2 N\rceil)\rceil + 7$, G is the number of common channels of two users.

III. THE GENERIC CH SEQUENCES

In this section, we introduce the generic multi-radio CH sequences that have bounded MTTR in CRNs. The MTTR bound will be further tightened in Section IV by optimizing a couple of parameters.

A. Complete symmetrization mapping

As mentioned in the Introduction section, the idea is to emulate each radio of a user as it were two radios in an *interval*. In the setting with two radios, suppose user 1 selects two channels a_1 and a_2 and user 2 selects two channels b_1 and b_2 in a time slot. To emulate that with a single radio, one needs the following four combinations (a_1, b_1), (a_1, b_2), (a_2, b_1) and (a_2, b_2) to occur in an interval. As such, the length of an interval is at least four time slots (if the clocks are synchronized). If the clocks are not synchronized, it will take longer. In the following, we introduce a class of codewords, called complete symmetrization class, that makes sure that these four combinations occur in an interval of M time slots.

Definition 1 (Complete symmetrization mapping) *Consider a set of M-bit codewords*

$$\{\mathbf{w}_i = (w_i(0), w_i(1), \ldots, w_i(M-1)), i = 1, 2, \ldots, K\}.$$

Let

$$\begin{aligned}&\textit{Rotate}(\mathbf{w}_i, d)\\ &= (w_i(d), w_i(d+1), \ldots, w_i((d+M-1) \textit{ mod } M))\\ &= (\tilde{w}_i(0), \tilde{w}_i(1), \ldots, \tilde{w}_i(M-1)),\end{aligned}$$

ALGORITHM 1: The Manchester mapping algorithm

Input: An integer $0 \le x \le 2^L - 1$.
Output: An M-bit codeword $\big(w(0), w(1), \dots, w(M-1)\big)$ with $M = 2 * L + 10$.
1: Let $\big(\beta_1(x), \beta_2(x), \dots, \beta_L(x)\big)$ be the binary representation of x, i.e., $x = \sum_{i=1}^{L} \beta_i(x) 2^{i-1}$.
2: Use the Manchester encoding scheme to encode x into a $2L$-bit codeword, $\big(\beta_1(x), \bar{\beta}_1(x), \beta_2(x), \bar{\beta}_2(x), \dots, \beta_L(x), \bar{\beta}_L(x)\big)$, where $\bar{\beta}_i(x)$ is the (binary) inverse of $\beta_i(x)$.
3: Add the 10-bit delimiter 0100011101 in front of the $2L$-bit codeword to form a $(2L+10)$-bit codeword.

be the vector obtained by cyclically shifting the vector $\mathbf{w}_i$ d *times. Then this set of codewords is called a* complete M-symmetrization class *if either the time shift* (d *mod* M) $\neq 0$ *or* $i \neq j$*, there exist* $0 \le \tau_1, \tau_2, \tau_3, \tau_4 \le M-1$ *such that*

(i) $(w_i(\tau_1), \tilde{w}_j(\tau_1)) = (0, 0)$,
(ii) $(w_i(\tau_2), \tilde{w}_j(\tau_2)) = (1, 1)$,
(iii) $(w_i(\tau_3), \tilde{w}_j(\tau_3)) = (0, 1)$*, and*
(iv) $(w_i(\tau_4), \tilde{w}_j(\tau_4)) = (1, 0)$.

A one-to-one mapping from the set of integers $[1, \dots, K]$ *to a complete* M*-symmetrization class is called a* complete M-symmetrization mapping.

We note that if $i = j$ and (d mod M) $= 0$, then $w_i(\tau) = \tilde{w}_i(\tau)$ for all τ and thus conditions (i) and (ii) hold trivially. As such, we know that conditions (i) and (ii) always hold for a complete symmetrization mapping.

In Algorithm 1, we show how one can construct a complete M-symmetrization mapping by using the Manchester coding (that replaces a bit 0 by the two bits 01 and a bit 1 by the two bits 10). From Algorithm 1, we have for an integer $0 \le x \le 2^L - 1$,

$$\begin{aligned}&(w_x(0), w_x(1), \dots, w_x(M-1)) \\ &= (0,1,0,0,0,1,1,1,0,1, \beta_1(x), \bar{\beta}_1(x), \\ &\qquad \beta_2(x), \bar{\beta}_2(x), \dots, \beta_L(x), \bar{\beta}_L(x)),\end{aligned}$$

where $\big(\beta_1(x), \beta_2(x), \dots, \beta_L(x)\big)$ is the binary representation of x. In the following lemma, we show that the Manchester mapping in Algorithm 1 is indeed a complete symmetrization mapping.

Lemma 2 *Algorithm 1 is a complete symmetrization mapping from* $x \in [0, 1, \dots, 2^L - 1]$ *to an* M*-bit codeword* $\big(w_x(0), w_x(1), \dots, w_x(M-1)\big)$ *with* $M = 2L + 10$.

Proof. From the Manchester mapping in Algorithm 1, we know that the substring of 3 consecutive 0's only appears in the 10-bit delimiter 0100011101 and thus it appears exactly once in the M-bit codeword for any cyclic shift d. This also holds for the substring of 3 consecutive 1's. Now consider the codeword $\big(w_x(0), w_x(1), \dots, w_x(M-1)\big)$ and the cyclically shifted codeword $\big(w_y(d), w_y(d+1), \dots, w_y((M-1+d) \bmod M)\big)$.

Case 1. (d mod M) $= 0$ and $x \neq y$: In this case, the 10-bit delimiters of two M-bit codewords are aligned. Thus, the conditions (i) and (ii) in Definition 1 are satisfied with $\tau_1 = 0$ and $\tau_2 = 1$. Since $x \neq y$, their binary representations are different. Thus, there exists a k such that $\beta_k(x) \neq \beta_k(y)$. In view of the Manchester mapping, there exist $10 + 2k \le \tau_3, \tau_4 \le 10 + 2k + 1$ such that the conditions (iii) and (iv) in Definition 1 are satisfied.

Case 2. (d mod M) $= 3$:

In this case, the substring of 3 consecutive 0's in $\big(w_x(0), w_x(1), \dots, w_x(M-1)\big)$ is aligned with the substring of 3 consecutive 1's in $\big(w_y(d), w_y(d+1), \dots, w_y((M-1+d) \bmod M)\big)$, i.e.,

$$\begin{aligned}&\big(w_x(0), w_x(1), \dots, w_x(M-1)\big) \\ &= (0,1,0,0,0,1,1,1,0,1,*,*,\dots), \\ &\big(w_y(d), w_y(d+1), \dots, w_y((M-1+d) \bmod M)\big) \\ &= (0,0,1,1,1,0,1,*,*\dots).\end{aligned}$$

It is easy to verify that $\tau_1 = 0$, $\tau_2 = 6$, $\tau_3 = 2$ and $\tau_4 = 1$ for this case.

Case 3. (d mod M) $= M - 3$:

In this case, the substring of 3 consecutive 1's in $\big(w_x(0), w_x(1), \dots, w_x(M-1)\big)$ is aligned with the substring of 3 consecutive 0's in $\big(w_y(d), w_y(d+1), \dots, w_y((M-1+d) \bmod M)\big)$, i.e.,

$$\begin{aligned}&\big(w_x(0), w_x(1), \dots, w_x(M-1)\big) \\ &= (0,1,0,0,0,1,1,1,0,1,*,*,\dots), \\ &\big(w_y(d), w_y(d+1), \dots, w_y((M-1+d) \bmod M)\big) \\ &= (*,*,*,0,1,0,0,0,1,1,\dots).\end{aligned}$$

It is easy to verify that $\tau_1 = 3$, $\tau_2 = 9$, $\tau_3 = 4$ and $\tau_4 = 5$.

Case 4. (d mod M) $\notin \{0, 3, M-3\}$:

In this case, the substring of 3 consecutive 0's in $\big(w_x(0), w_x(1), \dots, w_x(M-1)\big)$ is neither aligned with the substring of 3 consecutive 0's nor aligned with the substring of 3 consecutive 1's in $\big(w_y(d), w_y(d+1), \dots, w_y((M-1+d) \bmod M)\big)$. Thus, we know that $2 \le \tau_1, \tau_3 \le 4$ and the conditions (i) and (iii) in Definition 1 are satisfied. Similarly, the substring of 3 consecutive 1's in $\big(w_x(0), w_x(1), \dots, w_x(M-1)\big)$ is neither aligned with the substring of 3 consecutive 0's nor aligned with the substring of 3 consecutive 1's in $\big(w_y(d), w_y(d+1), \dots, w_y((M-1+d) \bmod M)\big)$. Thus, we know that $5 \le \tau_2, \tau_4 \le 7$ and the conditions (ii) and (iv) in Definition 1 are satisfied.

■

B. Each user has exactly two channels and one radio

Now consider the two-user rendezvous problem with a common channel labelling of the N channels, indexed from 0 to $N-1$. Suppose that each user has exactly two available channels and one radio. Since there is a common channel labelling of the N channels, without loss of generality we assume that $c_i(0) < c_i(1)$, $i = 1$ and 2. Let

$$(\beta_1(z), \beta_2(z), \ldots, \beta_{\lceil \log_2 N \rceil}(z))$$

be the binary representation of $z \in [0, 1, \ldots, N-1]$, i.e.,

$$z = \sum_{i=1}^{\lceil \log_2 N \rceil} \beta_i(z) 2^{i-1}.$$

Note that in our notation the first bit is the least significant bit of the binary representation. Based on the available channel set, we assign user i an integer x_i with

$$x_i = \max\{k : \beta_k(c_i(0)) < \beta_k(c_i(1))\} - 1. \tag{3}$$

The integer $x_i + 1$ is the largest bit that the binary representations of the two available channels $c_i(0)$ and $c_i(1)$ differ. Note that $0 \leq x_i \leq \lceil \log_2 N \rceil - 1$ and thus the binary representation of x_i requires at most $\lceil \log_2(\lceil \log_2 N \rceil) \rceil$ bits. Such an assignment method was previously used in [7], [25]. In the following theorem, we improve the MTTR from $16(\lceil \log_2 \log_2 N \rceil + 1)$ in [7] to $2\lceil \log_2(\lceil \log_2 N \rceil) \rceil + 10$ when $n_i = 2$ for all i.

Theorem 3 *Suppose that the assumption in (1) holds and $n_i = 2$ for $i = 1, 2$. User i uses the Manchester complete symmetrization mapping in Algorithm 1 with the integer x_i in (3) to generate an M-bit codeword $\big(w_{x_i}(0), w_{x_i}(1), \ldots, w_{x_i}(M-1)\big)$ with $M = 2\lceil \log_2(\lceil \log_2 N \rceil) \rceil + 10$. At time t, user i hops on channel $c_i(0)$ (resp. $c_i(1)$) if $w_{x_i}(t \bmod M) = 0$ (resp. 1). Then both users rendezvous within M time slots.*

Proof. We have shown in Lemma 2 that Algorithm 1 is indeed a complete symmetrization mapping. If $x_1 \neq x_2$, it then follows from the assumption in (1) and the four conditions (i), (ii), (iii) and (iv) in Definition 1 for a complete symmetrization mapping that these two users rendezvous within M time slots. Thus, we only need to consider the case that $x_1 = x_2$.

If $x_1 = x_2 = k$ for some k, then $\beta_k(c_1(0)) = \beta_k(c_2(0)) = 0$ and $\beta_k(c_1(1)) = \beta_k(c_2(1)) = 1$. This implies that $c_1(0) \neq c_2(1)$ as their binary representations are different. Similarly, $c_1(1) \neq c_2(0)$. Thus, under the assumption in (1), we have either $c_1(0) = c_2(0)$ or $c_1(1) = c_2(1)$. Thus, the two conditions (i) and (ii) in Definition 1 imply that these two users rendezvous within M time slots. ■

ALGORITHM 2: The (simple) modular clock algorithm

Input: An available channel set $\mathbf{c} = \{c(0), c(1), \ldots, c(n-1)\}$ and a period $p \geq |\mathbf{c}|$.

Output: A CH sequence $\{X(t), t = 0, 1, \ldots\}$ with $X(t) \in \mathbf{c}$.

1: For each t, let $k = (t \bmod p)$.

2: If $k \leq |\mathbf{c}| - 1$, let $X(t) = c(k)$.

3: Otherwise, select $X(t)$ uniformly at random from the available channel set $\mathbf{c}$.

ALGORITHM 3: The emulation algorithm

Input: An available channel set $\mathbf{c} = \{c(0), c(1), \ldots, c(n-1)\}$ and the total number of channels in the CRN N.

Output: A CH sequence $\{X(t), t = 0, 1, \ldots\}$ with $X(t) \in \mathbf{c}$.

0: Partition time into intervals with each interval consists of $2M$ time slots, where $M = 2\lceil \log_2(\lceil \log_2 N \rceil) \rceil + 10$.

1: Select two primes $p_1 > p_0 \geq |\mathbf{c}|$.

2: For the t^{th} interval, selects the first (resp. second) channel according to the (simple) modular clock algorithm in Algorithm 2 at time t by using the prime p_0 (resp. p_1) as its input. Let $c_a(t)$ and $c_b(t)$ be these two selected channels.

3: If $c_a(t) = c_b(t)$, replace one of them by another channel in $\mathbf{c}$.

4: Order these two channels so that $c_a(t) < c_b(t)$.

5: Let $x(t) + 1$ be the largest bit that the binary representations of the two available channels $c_a(t)$ and $c_b(t)$ differ, i.e.,

$$x(t) = \max\{k : \beta_k(c_a(t)) < \beta_k(c_b(t))\} - 1.$$

6: Within the t^{th} interval, uses the Manchester complete symmetrization algorithm in Algorithm 1 with the integer $x(t)$ to generate an M-bit codeword $\big(w_{x(t)}(0), w_{x(t)}(1), \ldots, w_{x(t)}(M-1)\big)$ with $M = 2\lceil \log_2(\lceil \log_2 N \rceil) \rceil + 10$.

7: At the τ^{th} time slot in the t^{th} interval, output the channel $c_a(t)$ (resp. $c_b(t)$) if $w_{x(t)}(\tau \bmod M) = 0$ (resp. 1).

C. Emulating two radios by a single radio

The result in Theorem 3 enables us to emulate two radios by using a single radio. To do this, we partition the time into a sequence of intervals with each interval consisting of $2M$ time slots. The $2M$ time slots in an interval ensure that the overlap between an interval of a user and the corresponding interval of another user consists of at least M consecutive time slots even when the clocks of the two users are not synchronized. Within an interval, user i runs the channel hopping sequence in Theorem 3 for a pair of two channels in its available channel set. Thus, at the time scale of intervals, each user behaves as if it had two radios. The detailed emulation algorithm is shown in Algorithm 3.

To select the two channels in an interval, we follow the two-prime modular clock algorithm in [3]. Specifically, user i first selects two primes $p_{i,1} > p_{i,0} \geq n_i$. For the t^{th} interval, user i selects one channel according to the (simple) modular clock algorithm (see Algorithm 2) at time t by using the prime $p_{i,0}$ as its input period. It also selects the other channel by using the same algorithm and the other prime $p_{i,1}$. Replace the second channel by an arbitrary available channel if these two selected channels are identical.

ALGORITHM 4: The multiple radio algorithm

Input: An available channel set $\mathbf{c} = \{c(0), c(1), \ldots, c(n-1)\}$, the number of radios m and the total number of channels in the CRN N.

Output: m CH sequences with $\{X^{(k)}(t), , t = 0, 1, \ldots\}$, $k = 1, 2, \ldots, m$ for the k^{th} radio.

1: Assign the $|\mathbf{c}|$ channels in the round robin fashion to the m radios. Let $\mathbf{c}^{(k)}$ be the set of channels assigned to the k^{th} radio, $k = 1, 2, \ldots, m$.

2: For the k^{th} radio, construct the CH sequence by using the emulation algorithm in Algorithm 3 with the input $\mathbf{c}^{(k)}$ and N.

Theorem 4 *Suppose that the assumption in (1) holds. User i uses the emulation algorithm in Algorithm 3 to generate its CH sequence. Then both users rendezvous within $18Mn_1 \cdot n_2$ time slots, where*

$$M = 2\lceil \log_2(\lceil \log_2 N \rceil) \rceil + 10.$$

Proof. In Step 0 of the emulation algorithm (Algorithm 3), the interval length is set to be $2M$ so that there is an overlap of (at least) M *consecutive* time slots for Theorem 3 to hold even when the clocks of the two users are not synchronized. As a direct consequence of Theorem 3 and the Chinese Remainder Theorem, we know that user i and user j rendezvous within $2Mp_{i,1} \cdot p_{j,1}$ time slots when the assumption in (1) holds. As $p_{i,1}$ can be found in $(n_i, 3n_i]$ [28], the MTTR is then bounded above $18Mn_1 \cdot n_2$, where $M = 2\lceil \log_2(\lceil \log_2 N \rceil) \rceil + 10$ (with N being the total number of channels). ■

D. Multiple radios

Now we consider the multiple radio setting. Suppose that user i has $m_i \geq 1$ radios, $i = 1$ and 2. It is possible that $m_i = 1$ in this setting. We first divide the n_i available channels as evenly as possible to the m_i radios so that each radio is assigned with at most $\lceil n_i/m_i \rceil$ channels. Let $\mathbf{c}_i^{(k)}$ be the channel assigned to the k^{th} radio of user i. For the k^{th} radio of user i, construct the CH sequence by using the emulation algorithm in Algorithm 3 with the input $\mathbf{c}_i^{(k)}$ and N. The detailed algorithm is shown in Algorithm 4.

Theorem 5 *Suppose that the assumption in (1) holds. User i uses the multiple radio algorithm in Algorithm 4 to generate its CH sequence. Then both users rendezvous within $18M\lceil n_1/m_1 \rceil \cdot \lceil n_2/m_2 \rceil$ time slots, where*

$$M = 2\lceil \log_2(\lceil \log_2 N \rceil) \rceil + 10.$$

Proof. Let $\mathbf{c}_i^{(k)}$ be the set of channels assigned to the k^{th} radio of user i, $k = 1, 2, \ldots, m_i$, $i = 1$ and 2. Under the assumption in (1), we first argue by contradiction that there must exist some $1 \leq k_1^* \leq m_1$ and $1 \leq k_2^* \leq m_2$ such that

$$\mathbf{c}_1^{(k_1^*)} \cap \mathbf{c}_2^{(k_2^*)} \neq \phi.$$

To see this, suppose that for all $1 \leq k_1 \leq m_1$ and $1 \leq k_2 \leq m_2$,

$$\mathbf{c}_1^{(k_1)} \cap \mathbf{c}_2^{(k_2)} = \phi.$$

This then implies that

$$\cup_{k_1=1}^{m_1} \cup_{k_2=1}^{m_2} \left(\mathbf{c}_1^{(k_1)} \cap \mathbf{c}_2^{(k_2)} \right) = \phi.$$

As $\mathbf{c}_1 = \cup_{k_1=1}^{m_1} \mathbf{c}_1^{(k_1)}$ and $\mathbf{c}_2 = \cup_{k_2=2}^{m_2} \mathbf{c}_2^{(k_2)}$, we then reach a contraction to the assumption in (1).

Since the channels are assigned in the round robin fashion to the radios, we know that

$$|\mathbf{c}_1^{(k_1^*)}| \leq \lceil n_1/m_1 \rceil$$

and

$$|\mathbf{c}_2^{(k_2^*)}| \leq \lceil n_2/m_2 \rceil.$$

The MTTR result of this theorem then follows from the MTTR result in Theorem 4 by using the k_1^{*th} radio of user 1 and the k_2^{*th} radio of user 2. ■

To illustrate the construction of our CH sequences, let us consider a CRN with two users: SU_1 and SU_2. Suppose that there are $N = 6$ channels, $\{0, 1, 2, 3, 4, 5\}$, and each user has a single radio, i.e., $m_1 = m_2 = 1$. The available channels for SU_1 is $\{1, 3, 4\}$ and the available channels for SU_2 is $\{2, 3\}$. Thus, $n_1 = 3$ and $n_2 = 2$. As such, we can simply choose $p_{1,0}$ = 3, $p_{1,1}$ = 5, $p_{2,0}$ = 2, and $p_{2,1}$ = 3. In Fig. 1 and Fig. 2, we show the two selected channels for these two users in the t^{th} interval after Step 2 of Algorithm 3 and after Step 4 of Algorithm 3, respectively.

t^{th} interval		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SU_1	$C_a(t)$	1	3	4	1	3	4	1	3	4	1	3	4	1	3	4
	$C_b(t)$	1	3	4	r	r	1	3	4	r	r	1	3	4	r	r
SU_2	$C_a(t)$	2	3	2	3	2	3	2	3	2	3	2	3	2	3	2
	$C_b(t)$	2	3	r	2	3	r	2	3	r	2	3	r	2	3	r

r: a randomly chosen channel from the available channel set.

Fig. 1. The two selected channels in the t^{th} interval after Step 2 of Algorithm 3.

t^{th} interval		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
SU_1	$C_a(t)$	1	3	$\underline{3^*}$	1	3	$\underline{1}$	1	3	$\underline{r(3)}$	1	$\underline{1}$	$\underline{3}$	1	3	$\underline{r(3)}$
	$C_b(t)$	4*	4*	$\underline{4}$	$r(3)$	$r(4)$	$\underline{4}$	3	4	$\underline{4}$	$r(3)$	$\underline{3}$	$\underline{4}$	4	$r(4)$	$\underline{4}$
SU_2	$C_a(t)$	2	2*	2	$\underline{2}$	2	$\underline{r(2)}$	2	2*	2	$\underline{2}$	2	$\underline{r(2)}$	2	2*	2
	$C_b(t)$	3*	3	$r(3)$	$\underline{3}$	3	$\underline{3}$	3*	3	$r(3)$	$\underline{3}$	3	$\underline{3}$	3*	3	$r(3)$

*: a channel replaced by a randomly chosen channel from the available channel set.
Underline: reordering of the two channels (so that $c_a(t) < c_b(t)$).

Fig. 2. The two selected channels in the t^{th} interval after Step 4 of Algorithm 3.

In Fig. 3, we show the CH sequences of Algorithm 3 in the first interval. To illustrate the effect that the clocks of these two users are not synchronized, we assume that there

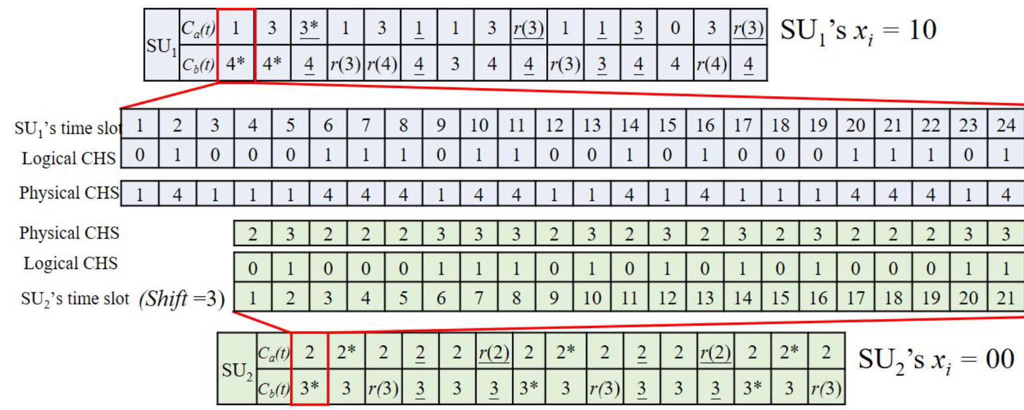


Fig. 3. An illustrating example of the CH sequences of Algorithm 3.

is a clock drift of three time slots between these two users. Since $N = 6$, we have $M = 2\lceil\log_2(\lceil\log_2 N\rceil)\rceil + 10 = 14$ and thus each interval contains $2M = 28$ times slots. In the first interval, SU_1 selects the two channels $c_a(1) = 1$ and $c_b(1) = 4$. The 3-bit binary representation of channel 1 (resp. 4) is $(\beta_1, \beta_2, \beta_3) = (1, 0, 0)$ (resp. $(0, 0, 1)$). As the largest bit that these two binary representations differ is the third bit, we have from (3) that $x_1 = 2$ (after Step 5 of Algorithm 3). The 2-bit binary representation of x_1 is $(1, 0)$ and the Manchester encoding of x_1 is $(1, 0, 0, 1)$. Adding the 10-bit delimiter 0100011101, the 14-bit codeword after the Manchester mapping in Algorithm 1 is 01000111011001 for SU_1 in the first time interval (after Step 6 of Algorithm 3). Thus, the 28 time slots in the first interval, called the logical CHS of SU_1, are labelled with 0100011101100101000111011001. Now every logical channel, labelled with 0 (resp. 1) in the 28 times slots of the first interval, is mapped to the physical channel 1 (resp. 4). This then leads to the physical channels hopping sequence 1411144414411414111444144114 in the first interval (after Step 7 of Algorithm 3). Similarly, SU_2 selects the two channels $c_a(1) = 2$ and $c_b(1) = 3$. The 3-bit binary representation of channel 2 (resp. 3) is $(\beta_1, \beta_2, \beta_3) = (0, 1, 0)$ (resp. $(1, 1, 0)$). As the largest bit that these two binary representations differ is the first bit, we have from (3) that $x_2 = 0$. The 2-bit binary representation of x_2 is $(0, 0)$ and the Manchester encoding of x_2 is $(0, 1, 0, 1)$. Thus, its logical CHS is 0100011101010101000111010101 and the corresponding physical CHS is 2322233323232323222333232323. Now in the second interval, SU_1 selects the two channels $c_a(2) = 3$ and $c_b(2) = 4$. These two users rendezvous at SU_2's 26^{th} time slot on channel 3 (see Fig. 3).

IV. The enhanced CH sequences

In this section, we discuss further improvements of our generic CH sequences in the previous section.

A. The minimum delimiter length for complete symmetrization mappings

In the Manchester mapping algorithm (Algorithm 1), the delimiter is 10 bits long. One natural question is whether one can further shorten the length of the delimiter. For $L = 2$, we test the complete symmetrization property for every bit string with the length less than 7 and the results are all negative. Thus, the length of the delimiter in the Manchester mapping algorithm is at least 7 bits long. For bit strings with the length 7, we only find two delimiters 1100101 and 0011010 that possess the complete symmetrization property. Note that the latter is simply the binary inverse of the former. With this finding from the exhaustive search for $L = 2$, we propose in Algorithm 5 the enhanced Manchester mapping algorithm. We prove in Lemma 6 below that Algorithm 5 is indeed a complete symmetrization mapping for any $L \geq 1$.

ALGORITHM 5: The enhanced Manchester mapping algorithm

The algorithm is the same as that in Algorithm 1 except Step 3.

3: Add the 7-bit delimiter 1100101 in front of the $2L$-bit codeword to form a $(2L + 7)$-bit codeword.

Lemma 6 *Algorithm 5 is a complete symmetrization mapping from $x \in [0, 1, \ldots, 2^L - 1]$ to an M-bit codeword $\big(w_x(0), w_x(1), \ldots, w_x(M-1)\big)$ with $M = 2L+7$ and $L \geq 1$.*

Proof.

As in the proof of Lemma 2, we consider the codeword $\big(w_x(0), w_x(1), \ldots, w_x(M-1)\big)$ and the cyclically shifted codeword $\big(w_y(d), w_y(d+1), \ldots, w_y((M-1+d) \bmod M)\big)$. Note from Algorithm 5, we have for an integer $0 \leq x \leq 2^L-1$,

$$
\begin{aligned}
&(w_x(0), w_x(1), \ldots, w_x(M-1)) \\
&= (1, 1, 0, 0, 1, 0, 1, \beta_1(x), \bar{\beta}_1(x), \\
&\qquad \beta_2(x), \bar{\beta}_2(x), \ldots, \beta_L(x), \bar{\beta}_L(x)),
\end{aligned}
$$

where $\big(\beta_1(x), \beta_2(x), \ldots, \beta_L(x)\big)$ is the binary representation of x.

Case 1. ($d \bmod M) = 0$ and $x \neq y$: In this case, the 7-bit delimiters of two M-bit codewords are aligned. Thus, the conditions (i) and (ii) in Definition 1 are satisfied with $\tau_1 = 2$ and $\tau_2 = 0$. Since $x \neq y$, their binary representations are different. Thus, there exists a k such that $\beta_k(x) \neq \beta_k(y)$. In view of the enhanced Manchester mapping, there exist $7 + 2k \leq \tau_3, \tau_4 \leq 7 + 2k + 1$ such that the conditions (iii) and (iv) in Definition 1 are satisfied.

Case 2. $(d \bmod M) = 1, 2$, and 3:

In this case, the four conditions are satisfied as shown in Fig. 4. In particular, when the shift is 1, we have $\tau_1 = 2$, $\tau_2 = 0$, $\tau_3 = 3$, and $\tau_4 = 1$. When the shift is 2, we have $\tau_1 = 3$, $\tau_2 = 4$, $\tau_3 = 2$, and $\tau_4 = 0$. Finally, when the shift is 3, we have $\tau_1 = 2$, $\tau_2 = 1$, $\tau_3 = 3$, and $\tau_4 = 0$.

t	0	1	2	3	4	5	6	7	8	9	10
$w_x(t)$	1	1	0	0	1	0	1	$\beta_1(x)$	$\tilde{\beta}_1(x)$	$\beta_2(x)$	$\tilde{\beta}_2(x)$
$w_y(t+1)$	1 τ_2	0 τ_4	0 τ_1	1 τ_3	0	1	$\beta_1(y)$	$\tilde{\beta}_1(y)$	$\beta_2(y)$	$\tilde{\beta}_2(y)$	…
$w_y(t+2)$	0 τ_4	0	1 τ_3	0 τ_1	1 τ_2	$\beta_1(y)$	$\tilde{\beta}_1(y)$	$\beta_2(y)$	$\tilde{\beta}_2(y)$	…	…
$w_y(t+3)$	0 τ_4	1 τ_2	0 τ_1	1 τ_3	$\beta_1(y)$	$\tilde{\beta}_1(y)$	$\beta_2(y)$	$\tilde{\beta}_2(y)$	…	…	…

Fig. 4. Illustration for the four conditions when $(d \bmod M) = 1, 2$, and 3.

Case 3. $(d \bmod M) = 4, 6, \ldots, 2L + 2$:

In this case, the four conditions are satisfied as shown in Fig. 5. Note that for every shift $k = (d \bmod M) = 4, 6, \ldots, 2L+2$, the 4-bit substring 1100 in the delimiter of codeword w_y is aligned with the 4-bit substring of codeword w_x

$$\beta_{\frac{2L+2-k}{2}}(x)\bar{\beta}_{\frac{2L+2-k}{2}}(x)\beta_{\frac{2L+4-k}{2}}(x)\bar{\beta}_{\frac{2L+4-k}{2}}(x).$$

Note that if $k = 2L+2$, we can simply define $\beta_0(x)\bar{\beta}_0(x)$ to be 01 as shown in Fig. 5. Since the four possible combinations of $\beta_{(2L+2-k)/2}(x)\bar{\beta}_{(2L+2-k)/2}(x)\beta_{(2L+4-k)/2}(x)\bar{\beta}_{(2L+4-k)/2}(x)$ are 0101, 0110, 1001, and 1010. The four conditions are satisfied.

t	…	5	6	7	8	9	10	…	$2L+3$	$2L+4$	$2L+5$	$2L+6$
$w_x(t)$	…	0	1	$\beta_1(x)$	$\tilde{\beta}_1(x)$	$\beta_2(x)$	$\tilde{\beta}_2(x)$	…	$\beta_{L-1}(x)$	$\tilde{\beta}_{L-1}(x)$	$\beta_L(x)$	$\tilde{\beta}_L(x)$
$w_y(t+4)$	…	$\beta_2(y)$	$\tilde{\beta}_2(y)$	…	…	…	…	…	1 $\boxed{\tau_2,\tau_3}$	1	0 $\boxed{\tau_1,\tau_4}$	0
…	…	…	…	…	…	…	…	…	…	…	…	…
$w_y(t+2L)$	…	$\beta_L(y)$	$\tilde{\beta}_L(y)$	1 $\boxed{\tau_2,\tau_3}$	1	0 $\boxed{\tau_1,\tau_4}$	0	…	…	…	…	…
$w_y(t+2L+2)$	…	1 τ_3	1 τ_2	0 $\boxed{\tau_1,\tau_4}$	0	1	0	…	…	…	…	…

Fig. 5. Illustration for four conditions when $(d \bmod M) = 4, 6, \ldots, 2L+2$.

Case 4. $(d \bmod M) = 5, 7, \ldots, 2L + 3$:

In this case, the four conditions are satisfied as shown in Fig. 6. Note that for every shift $k = (d \bmod M) = 5, 7, \ldots, 2L+3$, the 4-bit substring 1100 in the delimiter of codeword w_x is aligned with the 4-bit substring of codeword w_y

$$\beta_{(k-5)/2}(y)\bar{\beta}_{(k-5)/2}(y)\beta_{(k-6)/2}(y)\bar{\beta}_{(k-6)/2}(y).$$

If $(k-5)/2 = 0$, we can simply define $\beta_0(y)\bar{\beta}_0(y))$ to be 01 as shown in Fig. 6. Similar to the argument used in Case 3, the four conditions are satisfied.

t	0	1	2	3	4	5	6	7	8	9	10
$w_x(t)$	1	1	0	0	1	0	1	$\beta_1(x)$	$\tilde{\beta}_1(x)$	$\beta_2(x)$	$\tilde{\beta}_2(x)$
$w_y(t+5)$	0 τ_2	1 τ_4	$\beta_1(y)$ $\boxed{\tau_1,\tau_3}$	$\tilde{\beta}_1(y)$	$\beta_2(y)$	$\tilde{\beta}_2(y)$	…	…	…	…	…
$w_y(t+7)$	$\beta_1(y)$ $\boxed{\tau_2,\tau_4}$	$\tilde{\beta}_1(y)$	$\beta_2(y)$ $\boxed{\tau_1,\tau_3}$	$\tilde{\beta}_2(y)$	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…	…
$w_y(t+2L+3)$	$\beta_{L-1}(y)$ $\boxed{\tau_2,\tau_4}$	$\tilde{\beta}_{L-1}(y)$	$\beta_L(y)$ $\boxed{\tau_1,\tau_3}$	$\tilde{\beta}_L(y)$	0	0	1	1	0	1	0

Fig. 6. Illustration for four conditions when $(d \bmod M) = 5, 7, \ldots, 2L+3$.

Case 5. $(d \bmod M) = 2L + 4, 2L + 5, 2L + 6$:

In this case, the four conditions are satisfied as shown in Fig. 7. In particular, when the shift is $2L+6$, we have $\tau_1 = 3$, $\tau_2 = 1$, $\tau_3 = 2$, and $\tau_4 = 4$. When the shift is $2L + 5$, we have $\tau_1 = 5$, $\tau_2 = 6$, $\tau_3 = 2$, and $\tau_4 = 4$. Finally, when the shift is $2L + 4$, we have $\tau_1 = 5$, $\tau_2 = 4$, $\tau_3 = 3$, and $\tau_4 = 6$. ■

t	0	1	2	3	4	5	6	7	8	9	10
$w_x(t)$	1	1	0	0	1	0	1	$\beta_1(x)$	$\tilde{\beta}_1(x)$	$\beta_2(x)$	$\tilde{\beta}_2(x)$
$w_y(t+2L+6)$	$\tilde{\beta}_L(y)$	1 τ_2	1 τ_3	0 τ_1	0 τ_4	1	0	1	$\beta_1(y)$	$\tilde{\beta}_1(y)$	$\beta_2(y)$
$w_y(t+2L+5)$	$\beta_L(y)$	$\tilde{\beta}_L(y)$	1 τ_3	1	0 τ_4	0 τ_1	1 τ_2	0	1	$\beta_1(y)$	$\tilde{\beta}_1(y)$
$w_y(t+2L+4)$	$\tilde{\beta}_{L-1}(y)$	$\beta_L(y)$	$\tilde{\beta}_L(y)$	1 τ_3	1 τ_2	0 τ_1	0 τ_4	1	0	1	$\beta_1(y)$

Fig. 7. Illustration for four conditions when $(d \bmod M) = 2L + 4, 2L + 5, 2L + 6$

As a direct result of the complete symmetrization mapping property in Lemma 6 and the proof of Theorem 3, we have the following corollary for the two-user rendezvous problem in which each user has exactly two available channels and one radio.

Corollary 7 *Suppose that the assumption in (1) holds and $n_i = 2$ for $i = 1, 2$. User i uses the enhanced Manchester coding algorithm in Algorithm 5 with the integer x_i in (3) to generate an M-bit codeword $\big(w_{x_i}(0), w_{x_i}(1), \ldots, w_{x_i}(M-1)\big)$ with $M = 2\lceil \log_2(\lceil \log_2 N \rceil) \rceil + 7$. At time t, user i hops on channel $c_i(0)$ (resp. $c_i(1)$) if $w_{x_i}(t \bmod M) = 0$ (resp. 1). Then both users rendezvous within M time slots.*

B. The minimum interval length

In Step 0 of the emulation algorithm (Algorithm 3), the interval length is set to be $2M$ so that there is an overlap of (at least) M *consecutive* time slots for Theorem 3 to hold even when the clocks of the two users are not synchronized. We will show that the condition for these M time slots to be consecutive is not needed if we use the enhanced Manchester mapping algorithm in Algorithm 5. Specifically, we propose an enhanced emulation algorithm in Algorithm 6 to emulate two radios by a single radio. In Algorithm 6, we partition the time into a sequence of intervals with each interval consisting of only M time slots. The rest of the algorithm is the same as the original emulation algorithm in Algorithm 3.

In the following theorem, we show an enhanced version of Theorem 4.

Theorem 8 *Suppose that the assumption in (1) holds and $N \geq 3$. User i uses the MTTR-enhanced emulation algorithm in Algorithm 6 to generate its CH sequence. Then both users rendezvous within $9Mn_1 \cdot n_2$ time slots, where*

$$M = 2\lceil \log_2(\lceil \log_2 N \rceil) \rceil + 7.$$

Proof. Consider two users, SU_1 and SU_2. Since each user chooses two primes in Algorithm 6, without loss of generality we may assume that SU_1 selects a prime p_1 that is different

ALGORITHM 6: The MTTR-enhanced emulation algorithm

The algorithm is the same as that in Algorithm 3 except Step 0 and Step 6.

0: Partition time into intervals with each interval consists of M time slots, where $M = 2\lceil \log_2(\lceil \log_2 N \rceil) \rceil + 7$.

6: Within the t^{th} interval, uses the enhanced Manchester complete symmetrization algorithm in Algorithm 5 with the integer $x(t)$ to generate an M-bit codeword $\big(w_{x(t)}(0), w_{x(t)}(1), \ldots, w_{x(t)}(M-1)\big)$ with $M = 2\lceil \log_2(\lceil \log_2 N \rceil) \rceil + 7$.

from another prime p_2 selected by SU_2, i.e., $p_1 \neq p_2$. According to the assumption in (1), there is a common available channel c. Suppose that channel c is the c_1^{th} (resp. c_2^{th}) channel in the channel available set of SU_1 (resp. SU_2). Then it follows from the modular clock algorithm in Algorithm 2 that channel c will be used as one of the two channels in the $(\ell_1 p_1 + c_1)^{th}$ interval of SU_1 for $\ell_1 = 0, 1, 2, \ldots$. Similarly, channel c will be used as one of the two channels in the $(\ell_2 p_2 + c_2)^{th}$ interval of SU_2 for $\ell_2 = 0, 1, 2, \ldots$. Let d be the clock drift between these users. Also let $d_1 = \lfloor d/M \rfloor$ and $d_0 = (d \bmod M)$ so that $d = d_1 M + d_0$. Since p_1 and p_2 are two different primes, it follows from the Chinese Remainder Theorem that there exists a unique $0 \leq t_1 \leq p_1 p_2 - 1$ that satisfies the following equation:

$$t_1 = \ell_1 p_1 + c_1 = (\ell_2 + d_1) p_2 + c_2.$$

Then the first $M - d_0$ time slots of the $t_1{}^{th}$ interval of SU_1 overlaps with the last $M - d_0$ time slots of the $(t_1 - d_1)^{th}$ interval of SU_2. On the other hand, there also exists a unique $0 \leq t_2 \leq p_1 p_2 - 1$ that satisfies the following equation:

$$t_2 = \ell_1 p_1 + c_1 = (\ell_2 + d_1) p_2 + ((c_2 - 1) \bmod p_2).$$

Then the last d_0 time slots of the $t_2{}^{th}$ interval of SU_1 overlaps with the first d_0 time slots of the $(t_2 - d_1)^{th}$ interval of SU_2. In Fig. 8, we illustrate this by an example with $p_1 = 3$, $p_2 = 2$, $c_1 = 2$, $c_2 = 2$, and $d_1 = 0$. In this example, we have $t_1 = 2$ and $t_2 = 5$. We will show that these two SUs will rendezvous either in the $t_1{}^{th}$ interval or the $t_2{}^{th}$ of SU_1. As shown in Theorem 4, $p_1 \leq 3n_1$ and $p_2 \leq 3n_2$. The MTTR is then bounded above $9Mn_1 \cdot n_2$, where $M = 2\lceil \log_2(\lceil \log_2 N \rceil) \rceil + 7$.

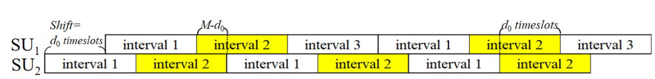


Fig. 8. An illustrating example of the overlapping time slots of a common available channel.

As in the proof of Lemma 6, we consider the following five cases. Let $L = \lceil \log_2(\lceil \log_2 N \rceil) \rceil$. Note that $M = 2L + 7$ and $L \geq 1$ (from $N \geq 3$).

Case 1. $d_0 = 0$:

In this case, the interval boundaries are aligned. As a direct consequence of Corollary 7, we know that these two SUs will rendezvous in the $t_1{}^{th}$ interval of SU_1.

Case 2. $d_0 = 1, 2,$ and 3:

In this case, the first $M - d_0$ time slots of the $t_1{}^{th}$ interval of SU_1 overlaps with the last $M - d_0$ time slots of the $(t_1 - d_1)^{th}$ interval of SU_2. As shown in Case 2 of the proof for Lemma 6 (see Fig. 4), when $d_0 = 1$, we have $\tau_1 = 2$, $\tau_2 = 0$, $\tau_3 = 3$, and $\tau_4 = 1$. Since the number of overlapped time slots is $M - 1 \geq 6 \geq \max_{1 \leq i \leq 4} \tau_i + 1$, we know that these two SUs will rendezvous in this interval from the complete symmetrization mapping property.

When $d_0 = 2$, we have $\tau_1 = 3$, $\tau_2 = 4$, $\tau_3 = 2$, and $\tau_4 = 0$. Since the number of overlapped time slots is $M - 2 \geq 5 \geq \max_{1 \leq i \leq 4} \tau_i + 1$, we know that these two SUs will rendezvous in this interval.

Finally, when $d_0 = 3$, we have $\tau_1 = 2$, $\tau_2 = 1$, $\tau_3 = 3$, and $\tau_4 = 0$. Since the number of overlapped time slots is $M - 3 \geq 4 \geq \max_{1 \leq i \leq 4} \tau_i + 1$, we know that these two SUs will rendezvous in this interval.

Case 3. $d_0 = 4, 6, \ldots, 2L + 2$:

In this case, the last d_0 time slots of the $t_2{}^{th}$ interval of SU_1 overlaps with the first d_0 time slots of the $(t_2 - d_1)^{th}$ interval of SU_2. Let x_1 (resp. x_2) be the codeword used by SU_1 (resp. SU_2) in the $t_2{}^{th}$ interval of SU_1 (resp. the $(t_2 - d_1)^{th}$ interval of SU_2). As shown in Case 3 of the proof for Lemma 6 (see Fig. 5), the 4-bit substring 1100 in the delimiter of codeword w_{x_2} is aligned with the 4-bit substring of codeword w_{x_1}

$$\beta_{\frac{2L+2-k}{2}}(x_1)\bar{\beta}_{\frac{2L+2-k}{2}}(x_1)\beta_{\frac{2L+4-k}{2}}(x_1)\bar{\beta}_{\frac{2L+4-k}{2}}(x_1).$$

As the 4-bit substring 1100 in the delimiter of codeword w_{x_2} appears at the first four bits of w_{x_2} and $d_0 \geq 4$, we know that these two SUs will rendezvous in this interval from the complete symmetrization mapping property.

Case 4. $d_0 = 5, 7, \ldots, 2L + 3$:

In this case, the first $M - d_0$ time slots of the $t_1{}^{th}$ interval of SU_1 overlaps with the last $M - d_0$ time slots of the $(t_1 - d_1)^{th}$ interval of SU_2. Let x_1 (resp. x_2) be the codeword used by SU_1 (resp. SU_2) in the $t_1{}^{th}$ interval of SU_1 (resp. the $(t_1 - d_1)^{th}$ interval of SU_2). As shown in Case 4 of the proof for Lemma 6 (see Fig. 6), the 4-bit substring 1100 in the delimiter of codeword w_{x_1} is aligned with the 4-bit substring of codeword w_{x_2}

$$\beta_{(k-5)/2}(x_2)\bar{\beta}_{(k-5)/2}(x_2)\beta_{(k-6)/2}(x_2)\bar{\beta}_{(k-6)/2}(x_2).$$

As the 4-bit substring 1100 in the delimiter of codeword w_{x_1} appears at the first four bits of w_{x_1} and $M - d_0 \geq 4$, we know that these two SUs will rendezvous in this interval from the complete symmetrization mapping property.

Case 5. $d_0 = 2L + 4, 2L + 5, 2L + 6$:

In this case, the last d_0 time slots of the $t_2{}^{th}$ interval of SU_1 overlaps with the first d_0 time slots of the $(t_2 - d_1)^{th}$ interval of SU_2. As shown in Case 5 of the proof for Lemma 6 (see Fig. 7), when $d_0 = 2L + 6$, we have $\tau_1 = 3$, $\tau_2 = 1$, $\tau_3 = 2$, and $\tau_4 = 4$. Since the number of overlapped time slots is $2L + 6 \geq M - \min_{1 \leq i \leq 4} \tau_i$, we know that these two SUs will rendezvous in this interval from the complete symmetrization mapping property.

When $d_0 = 2L + 5$, we have $\tau_1 = 5$, $\tau_2 = 6$, $\tau_3 = 2$, and $\tau_4 = 4$. Since the number of overlapped time slots is $2L + 5 \geq M - \min_{1 \leq i \leq 4} \tau_i$, we know that these two SUs will rendezvous in this interval.

Finally, when $d_0 = 2L+4$, we have $\tau_1 = 5$, $\tau_2 = 4$, $\tau_3 = 3$, and $\tau_4 = 6$. Since the number of overlapped time slots is $2L+4 \geq M - \min_{1\leq i\leq 4}\tau_i$, we know that these two SUs will rendezvous in this interval.

■

As an immediate result of Theorem 8, we have the following corollary for the MTTR for the multiple radio setting.

Corollary 9 *Suppose that the assumption in (1) holds and $N \geq 3$. Also, we modify Step 2 of the multiple radio algorithm in Algorithm 4 by using the MTTR-enhanced emulation algorithm in Algorithm 6. User i then uses the modified multiple radio algorithm to generate its CH sequence. Then both users rendezvous within $9M\lceil n_1/m_1\rceil \cdot \lceil n_2/m_2\rceil$ time slots, where*

$$M = 2\lceil \log_2(\lceil \log_2 N\rceil)\rceil + 7.$$

C. Improving the performance of ETTR

It is well-known that the simple random algorithm (that randomly chooses a channel from the available channel set in every time slot) performs very well in terms of ETTR in asynchronous heterogeneous CRNs. As such, to improve the performance of ETTR, it is suggested in [19] to make the rendezvous algorithms appear to be random. In Step 3 of the modular clock algorithm (Algorithm 2), we simply choose uniformly at random a channel, say channel c, from the available channel set. Such a random choice of a channel does not affect our proof for the MTTR bound. However, it does have a serious effect on the ETTR of our algorithm. This is because an interval in the MTTR-enhanced emulation algorithm (Algorithm 6) consists of M time slots, and channel c is used through the entire interval that selects channel c. To improve the ETTR of our algorithm, one should replace channel c in every time slot of the interval (that uses channel c) by another randomly selected channel. But this needs to be done delicately so that it won't affect the MTTR bound too much.

The idea is to add a fictitious channel, called channel N, in the CRN. By doing so, there are $N+1$ channels and channel N is not in the available channel set of any user. Now instead of choosing uniformly at random a channel from the available channel set in Step 3 of the modular clock algorithm, we simply choose the fictitious channel N (for further processing). In Algorithm 7, we propose the enhanced emulation algorithm that provides the detailed implementation of the idea. The enhanced emulation algorithm (Algorithm 7) uses the MTTR-enhanced emulation algorithm (Algorithm 6) with the additional modifications in Step 3 and Step 7 for the improvement of the ETTR performance. We also note that it is possible to accomplish this without adding the fictitious channel N. For instance, if channel $N-1$ is not in the available channel set of a user, then we can simply treat channel $N-1$ as the fictitious channel for that user. But it makes our presentation very complicated if channel $N-1$ is in the available channel set of a user. In that scenario, there are many cases that need to be treated separately. In the implementation of our enhanced algorithm in Section V, we do this without adding the fictitious channel N.

ALGORITHM 7: The enhanced emulation algorithm

Input: An available channel set $\mathbf{c} = \{c(0), c(1), \ldots, c(n-1)\}$ and the total number of channels in the CRN N.

Output: A CH sequence $\{X(t), t = 0, 1, \ldots\}$ with $X(t) \in \mathbf{c}$.

0: Partition time into intervals with each interval consists of M time slots, where $M = 2\lceil \log_2(\lceil \log_2(N+1)\rceil)\rceil + 7$.

1: Select two primes $p_1 > p_0 \geq |\mathbf{c}|$.

2: For the t^{th} interval, selects the first (resp. second) channel according to the (simple) modular clock algorithm in Algorithm 2 (with the fictitious channel N in Step 3 of the algorithm) at time t by using the prime p_0 (resp. p_1) as its input. Let $c_a(t)$ and $c_b(t)$ be these two selected channels.

3: If $c_a(t) = c_b(t)$, we consider the following two cases. If $c_a(t) = c_b(t) = N$, go directly to step 6. If $c_a(t) = c_b(t) \neq N$, we replace $c_b(t)$ by the fictitious channel N.

4: Order these two channels so that $c_a(t) < c_b(t)$.

5: Let $x(t)+1$ be the largest bit that the binary representations of the two available channels $c_a(t)$ and $c_b(t)$ differ, i.e.,

$$x(t) = \max\{k : \beta_k(c_a(t)) < \beta_k(c_b(t))\} - 1.$$

6: Within the t^{th} interval, uses the enhanced Manchester complete symmetrization algorithm in Algorithm 5 with the integer $x(t)$ to generate an M-bit codeword $\big(w_{x(t)}(0), w_{x(t)}(1), \ldots, w_{x(t)}(M-1)\big)$ with $M = 2\lceil \log_2(\lceil \log_2(N+1)\rceil)\rceil + 7$.

7: If $c_a(t) = c_b(t) = N$, we randomly select two different channels in every time slot of the interval from the channel available set $\mathbf{c}$. Otherwise, at the τ^{th} time slot in the t^{th} interval, we output the channel $c_a(t)$ (resp. $c_b(t)$) if $w_{x(t)}(\tau \bmod M) = 0$ (resp. 1). If one of the two output channels is the fictitious channel N, we randomly replace it by a channel from the available channel set $\mathbf{c}$ that is different from the other non-fictitious channel.

Another possible improvement of the ETTR of our algorithm is to change the starting interval of the CH sequence. If $p_0 > |\mathbf{c}|$, then it might be better to start from the $p_1 \cdot p_0 - (p_0 - |\mathbf{c}|)^{th}$ interval as Step 3 of the modular clock algorithm (Algorithm 2) will choose the fictitious channel N from the $p_1 \cdot p_0 - (p_0 - |\mathbf{c}|)^{th}$ interval onward.

V. Simulation Results

In this section, we conduct extensive simulations to compare the performance of our proposed algorithm (the enhanced version in Algorithm 7) with six commonly used multi-radio channel hopping algorithms in asynchronous heterogenous CRNs. We used an event-driven C++ simulator in our simulations to compare the performance with JS/I [5], JS/P [5], RPS [5], GCR [3], and AMRR [6]. The AMRR scheme provides options for the optimization of MTTR (AMRR/M) and ETTR (AMRR/E). To optimize MTTR, the AMRR scheme assigns half of the radios as stay radios. On the other hand, to optimize ETTR, the AMRR scheme assigns one radio as the stay radio. Since the GCR scheme assigns radios in pairs, the unpaired radio hops randomly on an available channel in every time slot when a user has an odd number of radios. When JS/P and JS/I jump to a channel that is not in the available channel set, it is replaced by a channel from the available channel set based on the local time.

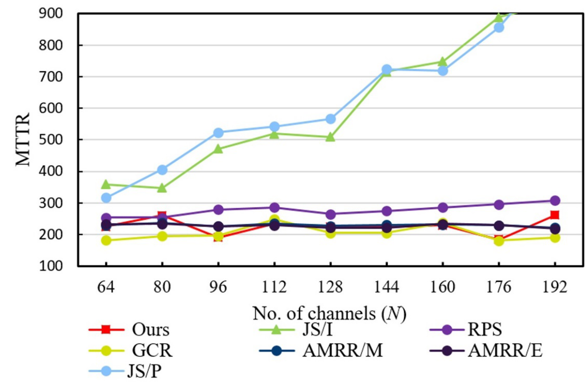
In our simulations, we assume that each SU has the information of the number of channels N in the CRN and its available channel set. Since GCR, RPS, AMRR are limited to the setting where the number of radios for each user has to be larger than one, the number of radios is set to be larger than one in our simulation (in order to compare our algorithm with these algorithms). To model the clock drift, each user randomly selects a (local) time to start its CH sequence. In our simulation, we only consider the rendezvous between two users. For each set of parameters, we generate 1,000 different available channel sets for the two users and perform 1,000 independent runs for each pair of the available channel sets. We then compute the maximum/average time-to-rendezvous as the measured MTTR/ETTR. The simulation results are obtained with 95% confidence intervals. Since the confidence intervals of ETTR's are all very small in our simulations, for clarity, we do not draw the confidence intervals in the figures.

A. Impact of the number of channels when the number of common channels is fixed

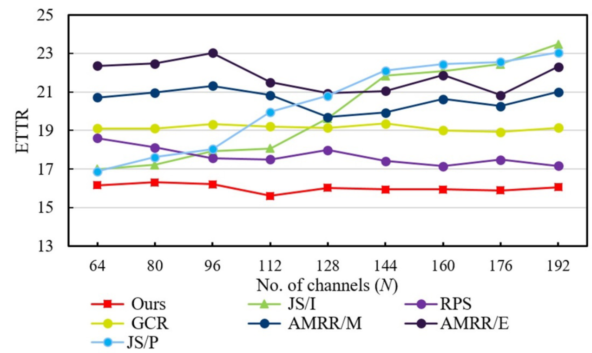
In this simulation, we vary the total number of channels N from 64 to 192 with fixed $n_1 = n_2$ uniformly chosen in $[14, 16]$, $m_1 = 2, m_2 = 4$, and the number of common channels $G = 2$. As shown in Fig. 9(a), GCR has the best performance in MTTR among all the schemes and the MTTR of our algorithm is almost the same as that of GCR. This is in line with the theoretical MTTR results in Table I as the MTTR of GCR is of the same order to ours. Moreover, the MTTR of our algorithm is almost invariant with respect to N in the range from 64 to 192. When n_1 and n_2 are fixed, the MTTR of our algorithm is only affected by M, the number of time slots in an interval, which is O($\log \log N$) (as stated in Corollary 9). On the other hand, the MTTRs of JS/P and JS/I are $O(N^3)$. As such, their MTTRs are worse than our algorithm when N is large. In Fig. 9(b), we show the comparison results for ETTR under the same setting. As shown in Fig. 9(b), our algorithm performs better than the other six schemes for any value of N. Moreover, the ETTR of our algorithm is also almost independent of N in the range from 64 to 192.

B. Impact of the number of channels when the number of common channels is proportional to the number of channels

In this simulation, we vary N from 64 to 192 and $n_1 = n_2 = N/4$, $G = N/8$ with fixed $m_1 = 2$ and $m_2 = 4$. In this simulation setting, the number of available channels

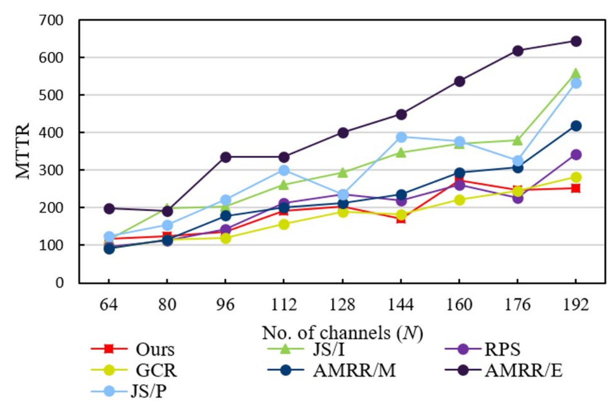


(a) MTTR vs. the number of channels N

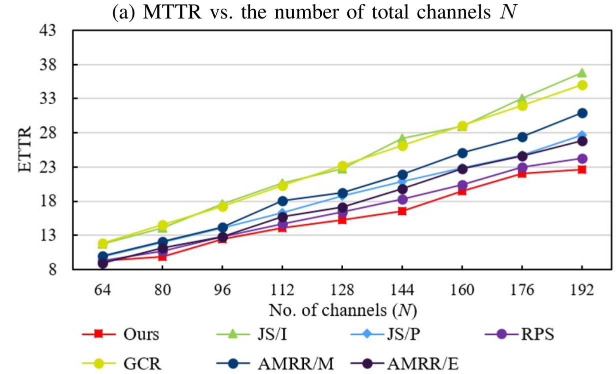


(b) ETTR vs. the number of channels N

Fig. 9. The effect of the number of channels on MTTR and ETTR with $n_1 = n_2$ uniformly chosen in $[14, 16]$, $G = 2$ and $m_1 = 2, m_2 = 4$.



(a) MTTR vs. the number of total channels N

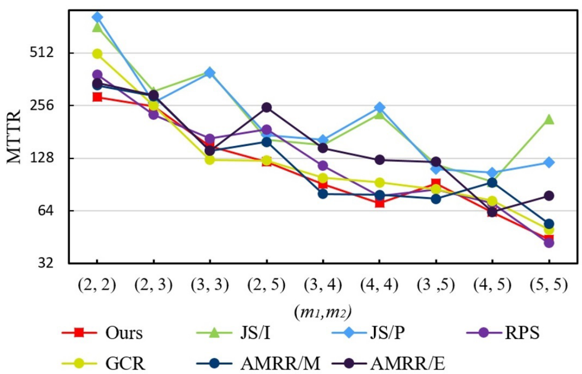


(b) ETTR vs. the number of total channels N

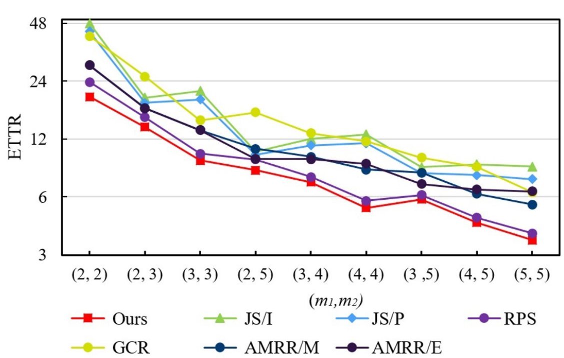
Fig. 10. The effect of the number of channels on MTTR and ETTR with $n_1 = n_2 = N/2$, $G = N/8$ and $m_1 = 3, m_2 = 6$.

(n_1, n_2) and the number of common channels (G) increase as the value of N increases. Since we increase the numbers of available channels n_1 and n_2 with respect to the total number of channels N, the MTTR bound for our algorithm in Theorem 5 is now $O(N^2 \log(\log N))$, which is still better than $O(N^3)$ for the MTTRs of JS/P and JS/I. In Fig. 10, we show the effect of the number of channels on MTTR and ETTR in this setting. As expected, the MTTR of AMRR/M is smaller than that of AMRR/E. Our MTTR is similar to RPS, GCR, and AMRR/M and better than AMRR/E, JS/P and JS/I in this setting. In Fig. 10(b), we can see that our algorithm has the best ETTR due to its random hopping property described in Section IV-C.

C. Impact of the number of radios



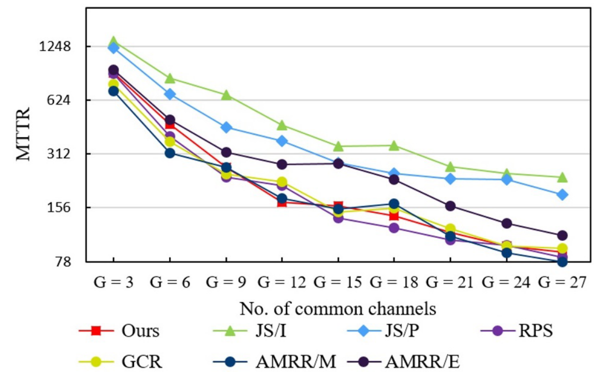
(a) MTTR for various settings of m_1 and m_2



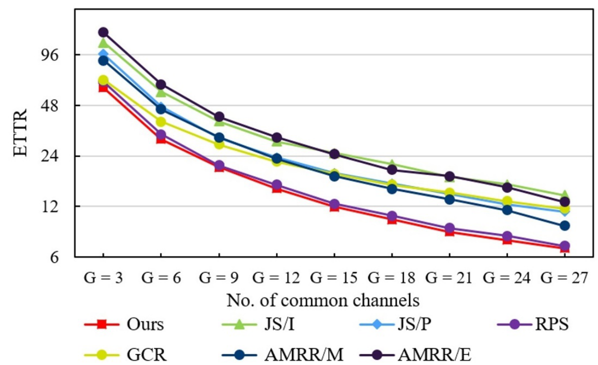
(b) ETTR for various settings of m_1 and m_2

Fig. 11. The effect of the number of radios on MTTR and ETTR for various settings of m_1 and m_2.

In this simulation, we fix N = 160, n_1 = n_2 = 40, G = 20. We then measure MTTR and ETTR for various settings of (m_1, m_2). The simulation results are shown in Fig. 11. As expected, both MTTR and ETTR decrease when the numbers of radios m_1 and m_2 are increased. The performance of AMRR is the worst for small values of m_1 or m_2. In JS/I and JS/P, when either m_1 or m_2 is a multiplicative factor of the other (ex: (2, 2), (2, 4)), the performance degrades as shown in Fig. 11. Finally, as shown in Fig. 11(b), our algorithm has the best ETTR for each case.



(a) MTTR vs. the number of common channels



(b) ETTR vs. the number of common channels

Fig. 12. The effect of the number of common channels on MTTR and ETTR for various common channels G with $n_1 = n_2$ = 64, $m_1 = m_2 = 5$.

D. Impact of the number of common channels

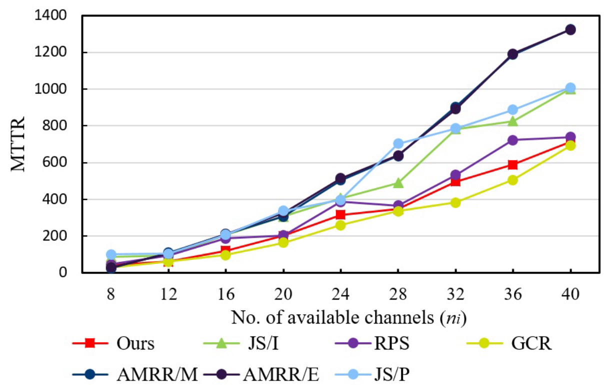
In this simulation, we fix $N = 160, n_1 = n_2 = 64, m_1 = 5, m_2 = 5$, and vary G from 3 to 27. The simulation results in Fig. 12 show that the MTTR of our algorithm is comparable to the lowest MTTR, and the ETTR of our algorithm is still the best among all the algorithms.

E. Impact of the number of available channels

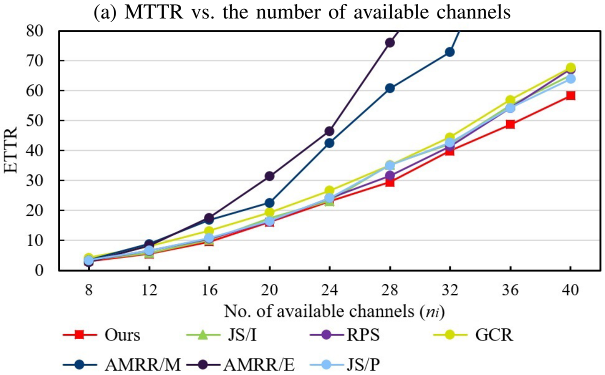
In this simulation, we vary n_1 and n_2 from 8 to 72. The number of channels $N = 160$ and $G = 3$. As the number of available channels increases, the MTTRs and the ETTRs of these algorithms also increase. This is because the number of common channels is fixed and both users need to explore more channels to rendezvous. Even though the MTTR of our algorithm is $O(n_1 n_2)$ (as stated in Corollary 9), the simulation results in Fig. 13 show that the MTTR and the ETTR of our algorithm are still very good among all the algorithms. In particular, we can observe from Fig. 13(a) that the MTTR of our algorithm is comparable to that of GCR and much smaller than the other algorithms. On the other hand, as shown in Fig. 13(b), the ETTR of our algorithm is still the best among all the algorithms.

VI. Conclusion

In this paper, we proposed a fast multi-radio rendezvous algorithm for a heterogeneous cognitive radio network. Our



(a) MTTR vs. the number of available channels



(b) ETTR vs. the number of available channels

Fig. 13. The effect of the number of available channels on MTTR and ETTR for various available channels with N = 160, $m_1 = 2, m_2$ = 5, and G =3.

algorithm is backward compatible and can also be used in the traditional setting where users are equipped with only one radio. For this, we introduced a new notion, called the complete symmetrization mapping, that allowed us to emulate two radios with a single radio in an interval. We showed for the two-user rendezvous problem in a CRN with N commonly labelled channels, the MTTR of our algorithm is bounded above by $9M\lceil n_1/m_1\rceil \cdot \lceil n_2/m_2\rceil$ time slots, where n_1 (resp. n_2) is the number of available channels to user 1 (resp. 2), m_1 (resp. m_2) is the number of radios for user 1 (resp. 2), and $M = 2\lceil \log_2(\lceil \log_2 N\rceil)\rceil + 7$. By conducting extensive simulations, we also showed that the MTTR of our algorithm is comparable to GCR [3] (that does not have a bounded MTTR for the setting with a single radio) and is better than the other schemes, including JS/I [5], JS/P [5], RPS [5], and AMRR [6]. The ETTR of our algorithm is the best among all these commonly used multi-radio algorithms in most parameter settings.

References

[1] C. S. Chang, Y.-C. Chang and J.-P. Sheu, "A Fast Multi-Radio Rendezvous Algorithm in Heterogeneous Cognitive Radio Networks," *Proceedings of the IEEE International Conference on Computer Communications (ICC)*, KANSAS CITY, MO, USA, May 20–24, 2018.

[2] S. Alpern and S. Gal. *The Theory of Search Games and Rendezvous*. Dordrecht: Kluwer Academic Publishers, 2003.

[3] G. Li, Z. Gu, X. Lin, H. Pu, and Q.-S. Hua, "Deterministic Distributed Rendezvous Algorithms for Multi-radio Cognitive Radio Networks," In *Proceedings of. ACM Proceedings of the 17th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pp. 313–320, 2014.

[4] N. C. Theis, R. W. Thomas, and L. A. DaSilva, "Rendezvous for Cognitive Radios," *IEEE Transactions on Mobile Computing*, vol. 10, no. 2, pp. 216–227, 2011.

[5] L. Yu, H. Liu, Y. W. Leung, X. Chu, and Z. Lin, "Multiple Radios for Fast Rendezvous in Cognitive Radio Networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 9, pp. 1917–1931, Sept. 2015.

[6] L. Yu, H. Liu, Y. W. Leung, X. Chu, and Z. Lin, "Adjustable Rendezvous in Multi-radio Cognitive Radio Networks," *Proceedings of the IEEE Global Communications Conference*, pp. 1–7, San Diego, CA, Dec 2015.

[7] Z. Gu, H. Pu, Q.-S. Hua, and F. C. M. Lau, "Improved Rendezvous Algorithms for Heterogeneous Cognitive Radio Networks," In *Proceedings of IEEE INFOCOM*, pp. 154–162, 2015.

[8] C.-S. Chang, D.-S. Lee, and W. Liao, "A Tutorial on Multichannel Rendezvous in Cognitive Radio Networks," Chapter 1 of *Cognitive Radio Networks: Performance, Applications and Technology*. Nova Science Publisher, 2018.

[9] Z. Gu, Y. Wang, Q.-S. Hua, and F. C. M. Lau, *Rendezvous in Distributed Systems: Theory, Algorithms and Applications*. Springer, 2017.

[10] C.-F. Shih, T. Y. Wu, and W. Liao, "DH-MAC: A Dynamic Channel Hopping MAC Protocol for Cognitive Radio Networks,"*Proceedings of the IEEE International Conference on Communications* (ICC), pp. 1–5, Cape Town, South Africa, May 2010.

[11] J. Li and J. Xie, "Practical Fast Multiple Radio Blind Rendezvous Schemes in Ad-Hoc Cognitive Radio Networks,"*Proceedings of Resilience Week* (RWS), pp. 1–6, Philadelphia, PA, USA, Aug. 2015.

[12] R. N. Yadav, R. Misra, "Periodic Channel-Hopping Sequence for Rendezvous in Cognitive Radio Networks,"*Proceedings of International Conference on Advances in Computing, Communications and Informatics* (ICACCI), pp. 1787–1792, Kochi, India, Aug. 2015.

[13] L. Yu, H. Liu, Y.-W. Leung, X. Chu, and Z. Lin, "Channel-Hopping Based on Available Channel Set for Rendezvous of Cognitive Radios," *Proceedings of the IEEE International Conference on Communications* (ICC), pp. 1573–1579, Sydney, NSW, June 2014.

[14] K. Bian and J.-M. Park, "Maximizing Rendezvous Diversity in Rendezvous Protocols for Decentralized Cognitive Radio Networks,"*IEEE Transactions on Mobile Computing*, Vol. 12, No. 7, pp. 1294–1307, July 2013.

[15] L. Chen, K. Bian, L. Chen, C. Liu, J.-M. J. Park, and X. Li, "A Group-Theoretic Framework for Rendezvous in Heterogeneous Cognitive Radio Networks,"*Proceedings of the fifteenth ACM international symposium on Mobile ad hoc networking and computing* (MobiHoc), pp. 165–174, Philadelphia, PA, USA, Aug. 2014.

[16] I. Chuang, H.-Y. Wu, K.-R. Lee, and Y.-H. Kuo, "A Fast Blind Rendezvous Method by Alternate Hop-and-Wait Channel Hopping in Cognitive Radio Networks,"*IEEE Transactions on Mobile Computing*, Vol. 13, No. 99, pp. 2171–2184, Jan. 2014.

[17] Z. Gu, Q.-S. Hua, and W. Dai, "Local Sequence Based Rendezvous Algorithms for Cognitive Radio Networks," *Proceedings of the IEEE 11th International Conference on Sensing, Communication, and Networking* (SECON), pp. 194–202, Singapore, July 2014.

[18] G.-Y. Chang, J.-F. Huang, and Y.-S. Wang, "Matrix-Based Channel Hopping Algorithms for Cognitive Radio Networks," *IEEE Transactions on Wireless Communications*, Vol. 14, No. 5, pp. 2755–2768, Jan. 2015.

[19] C. S. Chang, C.-Y. Chen, D.-S. Lee, and W. Liao, "Efficient Encoding of User IDs for Nearly Optimal Expected Time-To-Rendezvous in Heterogeneous Cognitive Radio Networks."*IEEE/ACM Transactions on Networking*, 2017.

[20] J. Li, H. Zhao, J. Wei, D. Ma, and L. Zhou, "Sender-Jump Receiver-Wait: a simple blind rendezvous algorithm for distributed cognitive radio networks," *IEEE Transactions on Mobile Computing*, Vol. 17, No.1, pp. 183–196. Jan. 2018.

[21] P. Bahl, R. Chandra and J. Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad Hoc Wireless Networks,"*Proceedings of the Annual International Conference on Mobile Computing and Networking* (MobiCom), pp. 216–230, NY, USA, Oct. 2004.

[22] P. K. Sahoo, S. Mohapatra, J.-P. Sheu, " Dynamic Spectrum Allocation Algorithms for Industrial Cognitive Radio Networks,"*IEEE Transactions on Industrial Informatics*, 2017.

[23] Y. R. Kondareddy and P. Agrawal, "Synchronized MAC Protocol for Multi-hop Cognitive Radio Networks," *Proceedings of the IEEE International Conference on Computer Communications (ICC)*, 2008.

[24] H. Liu, Z. Lin, X. Chu, and Y.-W. Leung, "Jump-Stay Rendezvous Algorithm for Cognitive Radio Networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 23, No. 10, pp. 1867–1881, Oct. 2012.

[25] S. Chen, A. Russell, A. Samanta, and R. Sundaram, "Deterministic blind rendezvous in cognitive radio networks." *IEEE 34th International*

Conference on Distributed Computing Systems (ICDCS), pp. 358–367, 2014.

[26] L. Chen, S. Shi, K. Bian, and Y. Ji, "Optimizing Average-Maximum TTR Trade-off for Cognitive Radio Rendezvous,"*Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 7707–7712, London, UK, June 2015.

[27] Z. Gu, H. Pu, Q.-S. Hua, and F. C. M. Lau, "Improved Rendezvous Algorithms for Heterogeneous Cognitive Radio Networks,"*Proceedings of the IEEE Conference on Computer Communications* (INFOCOM), pp. 154–162, Kowloon, Hong Kong, May 2015.

[28] M. El Bachraoui, "Primes in the interval [2n, 3n]," *Int. J. Contemp. Math. Sci.*, pp. 617–621, 2006.