A Fast Multi-Radio Rendezvous Algorithm in Heterogeneous Cognitive Radio Networks

Cheng-Shang Chang
Institute of Communications Engineering
National Tsing Hua University
Hsinchu 30013, Taiwan, R.O.C.
cschang@ee.nthu.edu.tw

Yeh-Cheng Chang
Department of Computer Science
National Tsing Hua University
Hsinchu 30013, Taiwan, R.O.C.
jas1123kimo@gmail.com

Jang-Ping Sheu
Department of Computer Science
National Tsing Hua University
Hsinchu 30013, Taiwan, R.O.C.
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 $2M$ time slots, where $M = 2\lceil \log_2(\lceil \log_2 N \rceil)\rceil + 10$. We show that the maximum time-to-rendezvous (MTTR) of our algorithm is bounded above by $18M\lceil n_1/m_1\rceil \cdot \lceil n_2/m_2\rceil$ 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 the expected time-to-rendezvous (ETTR) of our algorithm is also better than the two commonly used multi-radio algorithms, JS/Independent and JS/Parallel, in most parameter settings.**

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 [1] 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 the 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 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 [2]. 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 [3] 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 [2] , RPS [4], AMRR [5], MSS [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 $18M\lceil n_1/m_1\rceil \cdot \lceil n_2/m_2\rceil$ time slots, where $M = 2\lceil\log_2(\lceil\log_2 N\rceil)\rceil + 10$.
(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 the expected time-to-rendezvous (ETTR) of our algorithm is also better than the two commonly used multi-radio algorithms, JS/Independent and JS/Parallel [4], [8], in most parameter settings.

II. The CH sequences

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

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

```
ALGORITHM 1: The Manchester mapping algorithm
  Input: An integer $0 \le x \le 2^L - 1$.
  Output: An $M$-bit codeword
          $\big(w(0), w(1), \ldots, w(M-1)\big)$ with
          $M = 2 * L + 10$.
  1: Let $\big(\beta_1(x), \beta_2(x), \ldots, \beta_L(x)\big)$ 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,
     $\big(\beta_1(x), \bar\beta_1(x), \beta_2(x), \bar\beta_2(x), \ldots, \beta_L(x), \bar\beta_L(x)\big)$, 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.
```

Let

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

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 $(d \bmod M) \neq 0$ 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 $[1, \ldots, K]$ to a complete M-symmetrization class is called a *complete M-symmetrization mapping*.

We note that if $i = j$ and $(d \bmod M) = 0$, 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 $0 \le x \le 2^L - 1$,

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

where $\big(\beta_1(x), \beta_2(x), \ldots, \beta_L(x)\big)$ is the binary representation of x. 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 $x \in [0, 1, \ldots, 2^L - 1]$ to an M-bit codeword $\big(w_x(0), w_x(1), \ldots, w_x(M-1)\big)$ with $M = 2L + 10$.

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 $\big(w_x(0), w_x(1), \ldots, w_x(M-1)\big)$ and the cyclically shifted codeword $\big(w_y(d), w_y(d+1), \ldots, w_y((M-1+d) \bmod M)\big)$.

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 $10 + 2k \leq \tau_3, \tau_4 \leq 10 + 2k + 1$ 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 $\big(w_x(0), w_x(1), \ldots, w_x(M-1)\big)$ is aligned with the substring of 3 consecutive 1's in $\big(w_y(d), w_y(d+1), \ldots, w_y((M-1+d) \bmod M)\big)$, i.e.,

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

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

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

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 $\big(w_x(0), w_x(1), \ldots, w_x(M-1)\big)$ is neither aligned with the substring of 3 consecutive 0's nor aligned with the substring of 3 consecutive 1's in $\big(w_y(d), w_y(d+1), \ldots, w_y((M-1+d) \bmod M)\big)$. Thus, we know that $2 \leq \tau_1, \tau_3 \leq 4$ and the conditions (i) and (iii) in Definition 1 are satisfied. Similarly, the substring of 3 consecutive 1's in $\big(w_x(0), w_x(1), \ldots, w_x(M-1)\big)$ is neither aligned with the substring of 3 consecutive 0's nor aligned with the substring of 3 consecutive 1's in $\big(w_y(d), w_y(d+1), \ldots, w_y((M-1+d) \bmod M)\big)$. Thus, we know that $5 \leq \tau_2, \tau_4 \leq 7$ 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 $z \in [0, 1, \ldots, N-1]$. 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], [9]. 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. ■

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 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

```
ALGORITHM 2: The (simple) modular clock algorithm
Input: An available channel set
    $\mathbf{c} = \{c(0), c(1), \dots, c(n-1)\}$ and a period
    $p \geq |\mathbf{c}|$.
Output: A CH sequence $\{X(t), t = 0, 1, \dots\}$ 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
    $\mathbf{c} = \{c(0), c(1), \dots, c(n-1)\}$ and the total
    number of channels in the CRN $N$.
Output: A CH sequence $\{X(t), t = 0, 1, \dots\}$ with
    $X(t) \in \mathbf{c}$.
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 $p_1 > p_0 > |\mathbf{c}|$.
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
   $\big(w_{x(t)}(0), w_{x(t)}(1), \dots, w_{x(t)}(M-1)\big)$ with
   $M = 2\lceil \log_2(\lceil \log_2 N \rceil) \rceil + 10$.
7: At the $\tau^{th}$ 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).
```

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 [2]. 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
    $\mathbf{c} = \{c(0), c(1), \dots, c(n-1)\}$, the number of
    radios $m$ and the total number of channels in the
    CRN $N$.
Output: $m$ CH sequences with $\{X^{(k)}(t), , t = 0, 1, \dots\}$,
    $k = 1, 2, \dots, m$ 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, $k = 1, 2, \dots, m$.
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. 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 $(n_i, 3n_i]$ [10], 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, $k = 1, 2, \dots, m_i$, $i = 1$ 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

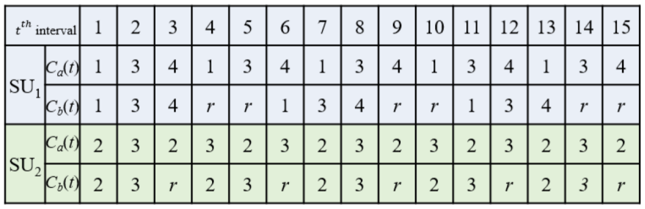
$$|\mathbf{c}_1^{(k_1^*)}| \leq \lceil n_1/m_1 \rceil$$

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. ■

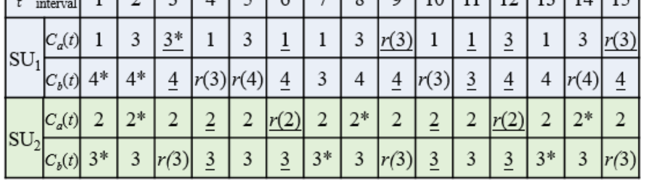
To illustrate the construction of our CH sequences, let us consider a CRN with two users: SU_1 and SU_2. Suppose that there are $N = 6$ channels, $\{0, 1, 2, 3, 4, 5\}$, and each user has a single radio, i.e., $m_1 = m_2 = 1$. The available channels for SU_1 is $\{1, 3, 4\}$ and the available channels 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 Line 2 of Algorithm 3 and after Line 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 Line 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 $c_a(t) < c_b(t)$).

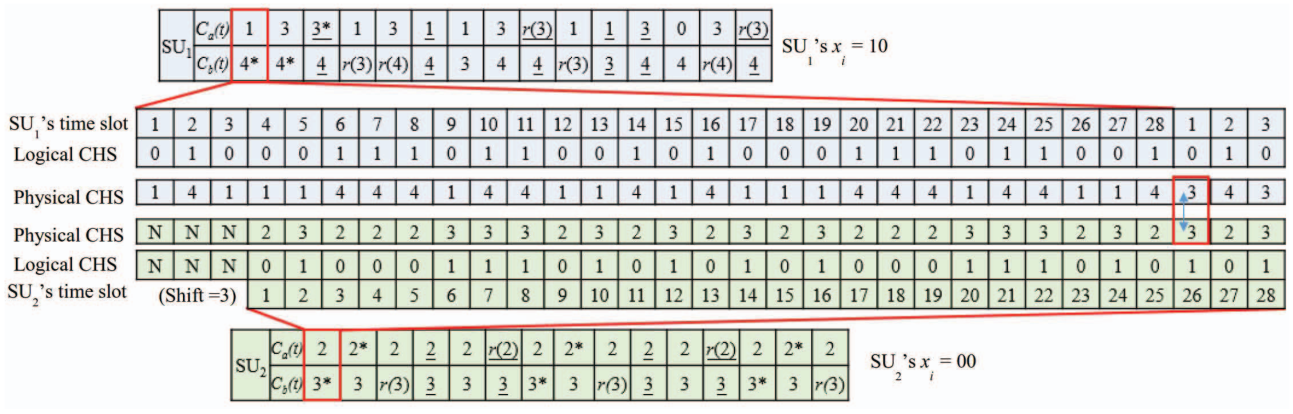
Fig. 2. The two selected channels in the t^{th} interval after Line 4 of Algorithm 3.

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 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 $(\beta_3, \beta_2, \beta_1) = (0, 0, 1)$ (resp. $(1, 0, 0)$). As the largest bit that these two binary representations differ is the third bit, we have from (3) that $x_1 = 2$ (after Line 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 Line 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 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 Line 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 $(\beta_3, \beta_2, \beta_1) = (0, 1, 0)$ (resp. $(0, 1, 1)$). 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).

III. Simulation Results

In this section, we conduct extensive simulations to compare the performance of our proposed algorithm with the two commonly used multi-radio channel hopping algorithms, JS/Independent and JS/Parallel [4], [8], by an event-driven C simulator. In [4], the MTTR of JS/Independent and JS/Parallel are $3NP(P - G) + 3P$, and $(3NP(P - G) + 3P)/m$, respectively, where G is the number of common channels between two users, P is a prime not smaller than N, and m is the number of radios when $m_1 = m_2$. Note that the MTTR of JS/Parallel is infinity when $m_1 \neq m_2$. There are many multi-radio schemes, e.g. GCR [2], RPS [4], AMRR [5], MSS [6], proposed in the literature. However, they are limited to the setting that the number of radios for each user has to be larger than 1. As such, they are not applicable for the traditional setting where some users might have only one radio. To model the clock drift, each user randomly selects a (local) time to start its CH sequence. 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.

In our experimental setup, we consider the rendezvous between two users. The number of common channels between two users is denoted by G. User 1 and user 2 are equipped with m_1 and m_2 radios, respectively. For each set of parameters, we generate 10,000 different available channel sets for these

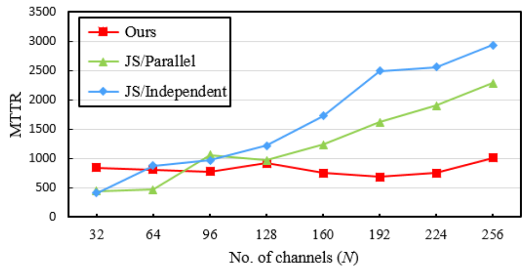


SU1 $C_a(t)$ 1 3 3* 1 3 1 1 3 r(3) 1 1 3 0 3 r(3)
$C_b(t)$ 4* 4* 4 r(3) r(4) 4 3 4 4 r(3) 3 4 4 r(4) 4
SU1's x_i = 10
SU1's time slot 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 1 2 3
Logical CHS 0 1 0 0 0 1 1 1 0 1 1 0 0 1 0 1 0 0 0 1 1 1 0 1 1 0 0 1 0 1 0
Physical CHS 1 4 1 1 1 4 4 4 1 4 4 1 1 4 1 4 1 1 1 4 4 4 1 4 4 1 1 4 3 4 3
Physical CHS N N N 2 3 2 2 2 3 3 3 2 3 2 3 2 3 2 3 2 2 2 3 3 3 2 3 2 3 2 3
Logical CHS N N N 0 1 0 0 0 1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0 1 0 1
SU2's time slot (Shift =3) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
SU2 $C_a(t)$ 2 2* 2 2 2 r(2) 2 2* 2 2 2 r(2) 2 2* 2
$C_b(t)$ 3* 3 r(3) 3 3 3 3* 3 r(3) 3 3 3 3* 3 r(3)
SU2's x_i = 00

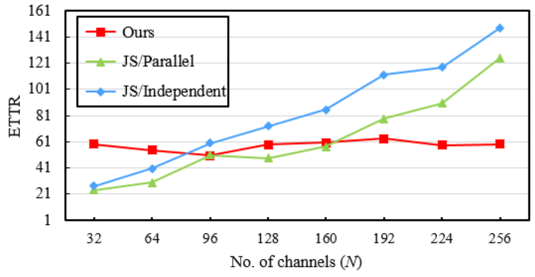
Fig. 3. An illustrating example of the CH sequences of Algorithm 3.

two users and perform 1,000 independent runs for each pair of the available channel sets. We then take the maximum/average time as MTTR/ETTR. The simulation results are obtained with 95% confidence interval. 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



(a) MTTR vs. the number of channels N

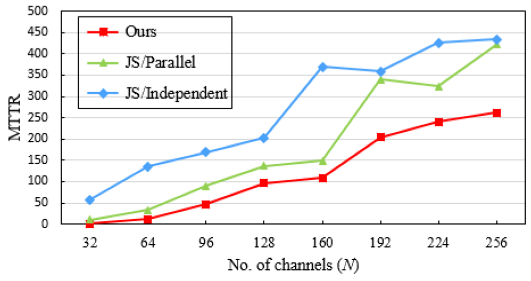


(b) ETTR vs. the number of channels N

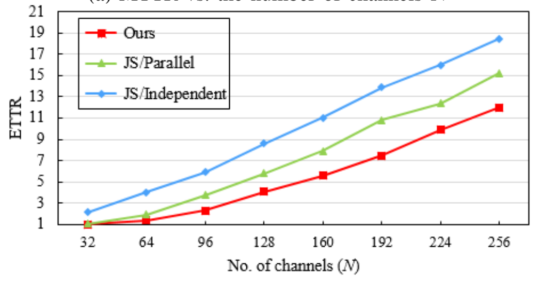
Fig. 4. The effect of the number of channels on MTTR and ETTR with $n_1 = n_2 = 16$, $G = 1$ and $m_1 = m_2 = 4$.

In this simulation, we vary N from 32 to 256 with fixed $n_1 = n_2 = 16$, $G = 1$ and $m_1 = m_2 = 4$. In Fig. 4(a), we can see that our algorithm has a much smaller MTTR than the other two schemes, JS/Parallel and JS/Independent, when the total number of channels N is larger than 96. Moreover, the MTTR of our algorithms is almost independent of N in the range from 32 to 256. 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 Theorem 5). On the other hand, the MTTRs of JS/Parallel and JS/Independent are $O(N^3)$. As such, their MTTRs are worse than our algorithm when N is large. In Fig. 4(b), we show the comparison results for ETTR under the same setting. Similarly, our algorithm performs better than the other two schemes when N is larger than 96. Moreover, the ETTR of our algorithms is also almost independent of N in the range from 32 to 256. Once again, this is because the number of time slots in an interval is $O(\log(\log N))$.

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



(a) MTTR vs. the number of channels N



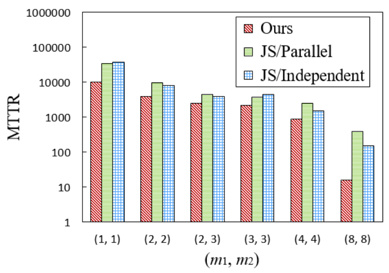
(b) ETTR vs. the number of channels N

Fig. 5. The effect of the number of channels on MTTR and ETTR with $n_1 = n_2 = N/8$, $G = N/16$ and $m_1 = m_2 = 4$.

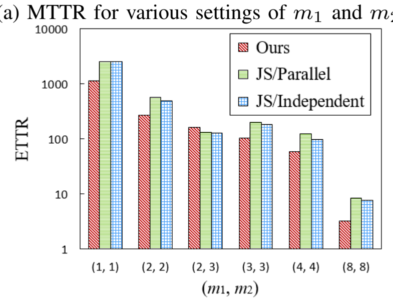
Instead of having a fixed number of common channels, we study the effect of the number of channels when the number of common channels is proportional to the number of channels. For this simulation, we vary N from 32 to 256 and set $n_1 =$

$n_2 = N/8, G = N/16$ with fixed m_1 = 4 and m_2 = 4. In Fig. 5, we show the effect of the number of channels on MTTR and ETTR in this setting. 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/Parallel and JS/Independent. As shown in Fig. 5, the MTTR and the ETTR of our algorithm are better than those of JS/Parallel and JS/Independent in this setting.

C. Impact of the number of radios



(a) MTTR for various settings of m_1 and m_2



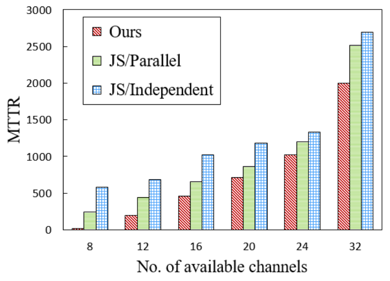
(b) ETTR for various settings of m_1 and m_2

Fig. 6. The effect of the number of radios on MTTR and ETTR for various settings of m_1 and m_2.

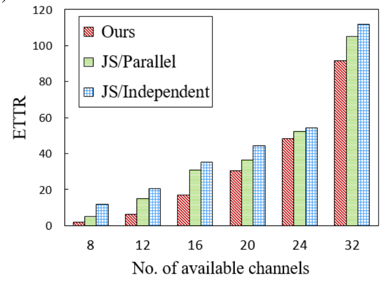
In this simulation, we fix N = 256, $n_1 = n_2$ =16 and G = 1. We then measure the MTTR and ETTR for various settings of (m_1, m_2). As expected, both MTTR and ETTR decreases when the numbers of radios m_1 and m_2 are increased. As stated in Theorem 5, the MTTR of our algorithm is $O(1/(m_1 m_2))$, which is much better than $O(1)$ for JS/Independent and $O(1/m)$ for JS/Parallel. The simulation results shown in Fig. 6 further verify that the MTTR and the ETTR of our algorithm are better than those of JS/Parallel and JS/Independent when the number of radios for each user is large.

D. Impact of the number of available channels

In this simulation, we fix $N = 256, m_1 = m_2 = 4$, and $G = 3$, and vary n_1 and n_2 (with $n_1 = n_2$) from 8 to 32. In Fig. 7, as the number of available channels increases, the MTTRs and the ETTRs of these three 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 Theorem 5), the simulation results in Fig. 7 show that the MTTR and the ETTR of our algorithm are still better than those of JS/Parallel and JS/Independent.



(a) MTTR vs. the number of available channels



(b)MTTR vs. the number of available channels

Fig. 7. The effect of the number of available channels on MTTR and ETTR for various available channels with N = 256, $m_1 = m_2$ = 4, and G =3.

References

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

[2] G. Li, Z. Gu, X. Lin, H. Pu, and Q.-S. Hua, "Deterministic distributed rendezvous algorithms for multi-radio cognitive radio networks," In *Proc. ACM Proceedings of the 17th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pp. 313-320, 2014.

[3] 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.

[4] 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.

[5] 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.

[6] B. Yang, W. Liang, M. Zheng, and Y. C. Liang, "Fully distributed channel-hopping algorithm for rendezvous setup in cognitive multiradio networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, pp. 8629–8643, Oct. 2016.

[7] Z. Gu, H. Pu, Q.-S. Hua, and F. C. M. Lau, "Improved rendezvous algorithms for heterogeneous cognitive radio networks," In *Proc. IEEE INFOCOM*, pp. 154–162, 2015.

[8] 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.

[9] 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.

[10] M. El Bachraoui, "Primes in the interval [2n, 3n]," *Int. J. Contemp. Math. Sci.*, Vol. 14, No. 9, pp. 1917-1931, Sep., 2015.