Efficient Encoding of User IDs for Nearly Optimal Expected Time-To-Rendezvous in Heterogeneous Cognitive Radio Networks

Cheng-Shang Chang, *Fellow, IEEE,* Cheng-Yu Chen, Duan-Shin Lee, *Senior Member, IEEE*, and Wanjiun Liao, *Fellow, IEEE*

***Abstract*—The multichannel rendezvous problem in cognitive radio networks (CRNs) has been a hot research topic lately. One of the most challenging settings of the multichannel rendezvous problem is the oblivious rendezvous problem in heterogeneous CRNs, where (i) there are no distinguishable roles of users, (ii) users' clocks are not synchronized, (iii) users may have different available channel sets, and (iv) there is no universal labelling of the channels. Most existing works in the literature focus on achieving deterministic bounds for the maximum conditional time-to-rendezvous (MCTTR) and perform poorly (in comparison with the random algorithm) for the expected time-to-rendezvous (ETTR) due to the "stay" modes in these works. In this paper, we tackle the oblivious rendezvous problem by taking both MCTTR and ETTR into consideration. In order to have guaranteed rendezvous, we only make two assumptions: (A1) there is at least one common available channel and (A2) there is a unique ID for each user. We first propose a new class of strong symmetrization mappings to encode user IDs for speeding up the rendezvous process. Two efficient and yet simple encoding schemes are proposed by utilizing the $\mathcal{C}$-transform and the existing 4B5B encoding. Based on the new class of strong symmetrization mappings, we propose the two-prime modular clock algorithm for the two-user rendezvous problem. The ETTR of our algorithm is almost the same as that of the random algorithm and its MCTTR is also comparable to the best existing bound. We also extend the two-prime modular clock algorithm for multiuser rendezvous by proposing the stick together algorithm and the spread out algorithm. One interesting finding for the multiuser rendezvous problem is that the spread out algorithm is not always better than the stick together algorithm as commonly claimed in the literature.**

***Index Terms*—rendezvous search, channel hopping, cognitive radio networks.**

I. Introduction

MOTIVATED by the recent development of cognitive radio networks (CRNs), the multichannel rendezvous problem has been studied extensively in the literature (see e.g., [1]–[33]). In a CRN, there are a set of frequency channels and two types of spectrum users: primary users and secondary users. Primary users have dedicated channels assigned to them. On the other hand, secondary users can only access channels that are not being blocked by primary users. As such, secondary users need to sense a number of frequency channels that are not blocked by primary users. Such a set of channels is called the *available channel set* for a secondary user. In order for a set of secondary users to communicate with each other, they need to find a channel that is commonly available to all of them. The multichannel rendezvous problem in a CRN is then for a set of secondary users to find a common available channel in a *distributed* manner by hopping over their available channels over time.

In this paper, we will simply call secondary users as users and assume time is partitioned into fixed length time slots (with synchronized slot boundaries as unsynchronized slot boundaries can be treated by doubling the length of time slots in the literature). A channel hopping (CH) algorithm of a user means an algorithm that selects a channel from its available channels in each time slot. When a set of users hop on a common available channel at the same time, we assume that there is a successful rendezvous and the set of users know such a channel can be used for setting up a communication link. Thus, the main objective of the multichannel rendezvous problem is to find CH algorithms that minimize the time-to-rendezvous (TTR) for a set of users to hop on a common available channel.

Most of the works in the literature considered the multichannel rendezvous problem with only two users and a universal labelling of channels. These works can be classified into various categories depending on their assumptions. A two-user rendezvous problem is called *asymmetric* (see e.g., ACH in [14] and ARCH in [24]) if one user can be identified as the *sender* and the other user can be identified as the *receiver*. On the other hand, a two-user rendezvous problem is called *symmetric* (see e.g., SSCH in [1], SYN-MAC in [4], QCH in [5] and DH-MAC in [6]) if there are no distinguishable roles of users. In general, the performance of asymmetric CH algorithms is better than that of symmetric CH algorithms as the sender and the receiver can use different algorithms to rendezvous in the asymmetric category. Also, a two-user rendezvous problem is *synchronous* if the indices of time slots of both users are the same. Synchronous CH algorithms can achieve better performance than asynchronous CH algorithms as both users have the same timing information. The symmetric and asynchronous category is considered the most challenging for the multichannel rendezvous problem. In the literature, there are several novel CH algorithms proposed for that setting with a universal labelling of channels (see e.g., SeqR [2], CRSEQ [7], DRSEQ [8], ASYNCH-ETCH [10], JS

C.-S. Chang, C.-Y. Chen, and D.-S. Lee are with the Institute of Communications Engineering, National Tsing Hua University, Hsinchu 300, Taiwan, R.O.C. email: cschang@ee.nthu.edu.tw, s100060021@m100.nthu.edu.tw, lds@cs.nthu.edu.tw.

W. Liao is with Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C. email: wjliao@ntu.edu.tw.

[11] and DRDS [16]). Two common worst-case performance metrics to evaluate these CH algorithms are (i) maximum time-to-rendezvous (MTTR): the maximum time for two users to rendezvous when *all the channels are available*, and (ii) maximum conditional time-to-rendezvous (MCTTR): the maximum time for two users to rendezvous when *there is at least one common available channel*. For comparisons of these CH algorithms for these two worst-case performance metrics, we refer to [22], [24]. There are also various lower bounds and optimal CH algorithms (that achieve lower bounds) for various categories in [28], [33].

Some recent works [9], [15], [17], [18], [18], [20], [21], [29] on the multichannel rendezvous problem considered heterogeneous CRNs, where users may have different available channel sets. Moreover, the assumption of a universal labelling of channels is removed in [19], [25], [26]. Such a multichannel rendezvous problem is called the *oblivious* rendezvous in heterogeneous CRNs in [19], [25], [26]. By using a unique ID assumption for each user as in [14], [22], it was shown in [19], [25], [26] that the MCTTR can still be upper bounded by a finite constant.

One performance metric that is not well studied in the previous works is the expected time-to-rendezvous (ETTR). Most of the CH algorithms in the literature perform rather poorly in terms of ETTR even when compared with the simple random algorithm. The rationale behind that is because there is usually a "stay" mode in these CH algorithms. When a user is in its "stay" mode, it stays in the same channel for a rather long period of time. As such, it is very likely that two users are in the "stay" mode and they stay in two different channels for a long period of time. In Section II, we will further illustrate this by relating it to a simple search problem. To address the large ETTR problem, a hybrid CH algorithm was proposed in [30] for a homogeneous CRN. The idea is to interleave the simple random algorithm with a periodic CH algorithm that has a bounded MCTTR, such as CRSEQ [7] and JS [11]. Interleaving is done by a periodic wake-up sequence that chooses the random algorithm when it is in the asleep mode and the CH algorithm when it is in the awake mode. Such a hybrid CH algorithm can greatly reduce the ETTR by decreasing the duty cycle of the wake-up sequence (so as to increase the chance to be in the asleep mode). However, the hybrid CH algorithm can only be used in a homogeneous CRN, where a universal labelling of channels is available. Motivated by this, our objective in this paper is to consider the *oblivious* rendezvous problem in heterogeneous CRNs and propose a CH algorithm such that its ETTR is comparable to that of the random algorithm while its MCTTR is still upper bounded by a finite constant (comparable to the best bound in the literature). In addition to the multichannel rendezvous problem with two users, we will also consider the scenario with multiple users in a heterogeneous cognitive radio network.

As the oblivious rendezvous problem in [19], [25], [26], we consider the most challenging symmetric and asynchronous category in heterogeneous CRNs, i.e., (i) there are no distinguishable roles of users, (ii) users' clocks are not synchronized, (iii) users may have different available channel sets, and (iv) there is no universal labelling of the channels available to the users. In order to have a guaranteed rendezvous, we make the following two assumptions in this paper:

(A1) There is at least one channel that is commonly available to the users. Specifically, suppose that there are K users and the available channel set for user i 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, $i = 1, 2, \ldots, K$. Then

$$\cap_{i=1}^{K} \mathbf{c}_i \neq \phi. \tag{1}$$

(A2) Each user is assigned with an L-bit unique ID (with $2^L \geq K$).

As we do not assume that the clocks of these K users are synchronized (to the global clock), every user operates on its local time. Denote by $X_i(t)$, $i = 1, 2, \ldots, K$, the channel selected by user i at its local time t. The time-to-rendezvous (TTR), denoted by T, is the first time that these K users select a common channel that is available to every user, i.e.,

$$T = \inf\{t : X_1(t + d_1) = X_2(t + d_2) = \ldots = X_K(t + d_K)\}, \tag{2}$$

where d_i, $i = 1, 2, \ldots, K$, are the clock drifts to the global clock.

Based on the two assumptions in (A1) and (A2), our contributions of this paper are as follows:

(i) We propose a new class of *strong symmetrization mappings* to encode user IDs for multichannel rendezvous. This class of mappings has nicer properties than the symmetrization mappings in [26] for speeding up the rendezvous process. By using the $\mathcal{C}$-transform in [34], [35], we show there exist strong symmetrization mappings with code rates arbitrarily close to 1. Utilizing the existing 4B5B encoding scheme, we also propose a strong symmetrization mapping with code rate near 80%.

(ii) Based on strong symmetrization mappings, we propose the two-prime modular clock algorithm in Algorithm 4 for oblivious rendezvous of two users. In the original modular clock algorithm, there is no guarantee that two users can rendezvous within a finite number of time slots (see e.g., Proposition 5 of [9]). Our two-prime modular clock algorithm fixes this problem and guarantees that the MCTTR for users i and j is upper bounded by $6Mn_in_j$ for the 4B5B encoding, where $M = (\lceil L/4 \rceil * 5 + 6)$. In particular, when $2^L = K$, the MCTTR upper bound is $O((\log_2 K)n_in_j)$, which is comparable to the Conversion Based Hopping (CBH) algorithm in [25] and the Advanced Rendezvous Protocol [26] (see Table I). Moreover, from our simulation results, its ETTR is almost the same as that of the random algorithm and is much better than other CH algorithms in the literature, including Modified Modular Clock Algorithm [9], FRCH [17], CBH [25] and Advanced Rendezvous Protocol [26].

(iii) We extend the two-prime modular clock algorithm for multiuser rendezvous by keeping a certain set of state information. To motivate the study of multiuser rendezvous, it was argued in [31] that a group of users need to rendezvous on the

TABLE I: MCTTR comparisons for oblivious rendezvous algorithms with the unique ID assumption

Algorithms	MCTTR
CBH [25]	$O(L\max[n_i n_j]^2)^{(\mathrm{i})}$
Adv. rdv [26]	$O(Ln_i n_j)^{(\mathrm{i})}$
Two-prime (this paper)	$O(Ln_i n_j)^{(\mathrm{i})}$

(i): L is a constant depending on the number of users K

same channel periodically so as to update their common time-dependent group key for secure communication. Such a time-dependent key update cannot be achieved by a sequence of pairwise rendezvous. Other applications and generalizations, including local broadcast and data aggregation, were addressed in [32]. As in [11], [19], when a group of users rendezvous on a channel, a leader is elected and both the state information and clocks are synchronized to those of the leader. The key challenge for the multiuser rendezvous is then how to adjust the CH sequences after a successful rendezvous of a subset of users so as to speed up the rendezvous process. We consider two different extensions: the stick together algorithm in Algorithm 5 and the spread out algorithm in Algorithm 6. In the stick together algorithm, all the users following a leader hop along with the leader. On the other hand, in the spread out algorithm, a user following a leader may hop on one of its available channels that is different from the channel selected by its leader. Our simulation results show that the ETTR of our stick together algorithm is still almost the same as that of the random algorithm. Moreover, the spread out algorithm is not always better than the stick together algorithm as commonly claimed in the literature (see e.g., [13]). In particular, when the number of common channels among a set of multiple users is very small, the spread out algorithm that hops on one of its available channels does not improve the rendezvous probability.

The rest of this paper is organized as follows: In Section II, we consider the two-user rendezvous problem. There we propose the strong symmetrization mappings, the 4B5B encoding and the two-prime modular clock algorithm. In Section III, we extend the two-prime modular clock algorithm for the multiuser rendezvous problem. In Section IV, we conduct extensive simulations to compare the performance of our two-prime modular clock algorithm with that of some best-performed channel hopping algorithms in the literature. Finally, we conclude the paper in Section V.

II. Two-user rendezvous

In this section, we consider the case with two users, i.e., $K = 2$. To gain some insights of the multichannel rendezvous problem, let us first consider a simple search problem. Suppose there are n items and only n_g items of these n items are good. The search problem is to minimize the time to find a good item. Clearly, if we examine an item and find out that it is not a good item, then this item should not be examined again (as it is a complete waste of time to do it again). As such, the optimal policy is to simply go through these n items one by one. As the n_g good times are uniformly distributed among any sequential search order of the n times, we can derive from the order statistics that the average time for the optimal policy to find a good item is $(n+1)/(n_g+1)$. Now we map the oblivious rendezvous problem with two users to this simple search problem. Consider users i and j. As there is no universal labelling of the channels, one can view each pair of available channels of these two users $(c_i(\tau_1), c_j(\tau_2))$ as an item and thus there are $n_i n_j$ items and $n_{i,j}$ good items, where $n_{i,j} = |\mathbf{c}_i \cap \mathbf{c}_j|$ is the number of common available channels between users i and j. This then leads to the following lower bound for the expected time-to-rendezvous (ETTR) for the multichannel rendezvous problem with two users.

Proposition 1: For the oblivious rendezvous problem, the ETTR for the two users i and j is lower bounded by $(n_i n_j + 1)/(n_{i,j}+1)$, where $n_{i,j}$ is the number of common available channels between users i and j.

For the oblivious rendezvous problem, each user learns nothing about the other user after an unsuccessful rendezvous. This is worse than the simple search problem where a bad item can be identified after an unsuccessful search. In view of this, a good CH algorithm for the oblivious rendezvous problem should avoid repeatedly hopping over the same channel pair even though the two users are not aware of whether they have hopped to a particular channel pair before. If n_i and n_j are relatively prime, this can be done perfectly by cycling through each available channel of a user over time (see the ϵ_1-rendezvous in [26]). In this case, the lower bound in Proposition 1 is achieved. Of course, the hard part is when n_i and n_j are not relatively prime.

We note that if there is a universal labelling of the channels, then it is possible for each user to learn something about the other user after an unsuccessful rendezvous. When the clocks of the two users are synchronized and all the channels are available, it was shown in [28] that finite projective planes can be used for each user to eliminate some choices of lines of the other user after an unsuccessful rendezvous. Such an approach was shown to be optimal in the sense of achieving the ETTR lower bound in the symmetric and synchronous setting in [28]. In the case that the clocks of the two users are not synchronized, difference sets can be used for eliminating some choices of the starting phase of the other user [8], [12], [16]. In particular, it was shown in Theorem 3 of [16] that the minimum period of a periodic CH sequence that has a bounded MCTTR for a system of N channels in the symmetric and asynchronous setting is at least N^2+N+1 for $N \geq 3$ and N is a prime power. Such a lower bound is achieved by a CH sequence for $N = 8$ in [12].

A simple random algorithm is considered in Algorithm 1 of [9]. For the random algorithm, each user randomly selects one of its available channels at time t. By doing so, the ETTR of the random algorithm is $n_i n_j / n_{i,j}$, which is very close to the lower bound in Proposition 1 (when $n_{i,j}$ is not too small). Most existing multichannel rendezvous algorithms (see e.g., [15], [17], [20], [21]) focus on MCTTR and they perform poorly in terms of ETTR when compared with the random algorithm. The rationale why these algorithms perform poorly is because there is usually a "stay" mode in these algorithms. When a user is in its "stay" mode, it stays in the same channel

for a rather long period of time. As such, it is very likely that two users are in the "stay" mode and they stay in two different channels for a long period of time. This corresponds to the scenario that a bad item is repeatedly examined for many times in the simple search problem.

Motivated by this, we will propose a CH algorithm that has no stay mode so as to reduce the possibility of hopping the same channel pair repeatedly. We will show that the MCTTR of our CH algorithm is $O(n_i n_j)$ and its ETTR is comparable to that of the random algorithm.

A. Deterministic modular clock algorithm

Algorithm 1 The deterministic modular clock algorithm

Require: An available channel set $\mathbf{c} = \{c(0), c(1), \ldots, c(n-1)\}$, a period $p \geq n$, a slope $r > 0$ that is relatively prime to p, and a bias $0 \leq b \leq p-1$.

Ensure: A deterministic sequence $\{\alpha(t), t = 0, 1, \ldots\}$ with $\alpha(t) \in \mathbf{c}$.

1: Let $z = 0$.
2: For each t, let $k = ((r * t + b) \bmod p)$.
3: If $k \leq n-1$, let $\alpha(t) = c(k)$.
4: Otherwise, let $\alpha(t) = c(z)$ and update $z \leftarrow ((z+1) \bmod n)$.

Our CH algorithm is inspired by the modular clock algorithm in [9]. In order to apply the modular clock algorithm for multiuser rendezvous, we remove the random part in the original modular clock algorithm in [9]. In Algorithm 1, we outline the *deterministic* modular clock algorithm. In addition to the available channel set, the algorithm needs three parameters: the period p that is an integer not smaller than the number of available channels n, the slope r that is relatively prime to p, and the bias that is an integer selected from $\{0, 1, \ldots, p-1\}$. If the clock k in Line 3 of the algorithm is not greater than $n-1$, then a channel is selected from the available channel set by using the clock as the index. Otherwise, a channel in the available channel set is selected in a round-robin fashion according to the pointer z (to mimic a random selection).

In the following lemma, we show two important properties of the deterministic modular clock algorithm. The proof can be easily done by using the Chinese Remainder Theorem and thus omitted due to space limitation.

Lemma 2: Suppose that user i generates the sequence $\{\alpha_i(t), t \geq 0\}$ by using the deterministic modular clock algorithm in Algorithm 1 with the available channel set $\mathbf{c_i} = \{c_i(0), c_i(1), \ldots, c_i(n_i-1)\}$, the period $p_i \geq n_i$, the slope $r_i > 0$ that is relatively prime to p_i, and the bias $0 \leq b_i \leq p_i - 1$.

(i) For any integer d, $\mathbf{c_i} \subset \{\alpha_i(t+d), t = 0, 1, \ldots, p_i - 1\}$.

(ii) Let $\mathbf{c_i} \times \mathbf{c_j} = \{(c_i(\tau_1), c_j(\tau_2)), \tau_1 = 0, 1, \ldots, n_i - 1, \tau_2 = 0, 1, \ldots, n_j - 1\}$ be the set that contains all the pairs of available channels for users i and j. If p_i and p_j are relatively prime, then $\mathbf{c_i} \times \mathbf{c_j} \subset \{(\alpha_i(t+d_1), \alpha_j(t+d_2)), t = 0, 1, \ldots, p_i p_j - 1\}$ for any integers d_1 and d_2.

From Lemma 2(ii), we have the following corollary.

Corollary 3: Suppose that user i (resp. user j) uses the deterministic modular clock algorithm in Algorithm 1 to generate its CH sequence with the period p_i (resp. p_j). If p_i and p_j are relatively prime, then under (A1) these two users will rendezvous on every common available channel at least once within $p_i p_j$ time slots.

We note that Corollary 3 was previous shown in Theorem 4 of [9] when p_i and p_j are assumed to be two different primes. As shown in [9], the problem arises when p_i and p_j are not relatively prime. In this case, there is no guarantee that these two users can rendezvous within a finite number of time slots (see e.g., Proposition 5 of [9]). To address such a problem, it is suggested in [9] that user i randomly selects a prime in $[n_i, 2n_i]$. Then it was shown numerically that the probability for two users to select the same prime is very small when the number of available channels for each user is large. But this still does not guarantee that two users can rendezvous within a finite number of time slots.

B. Strong symmetrization mapping

To ensure that two users can select two co-prime periods for the deterministic modular clock algorithm, one common trick (see e.g., [14], [22], [26], [33]) is to use the unique ID property in (A2) to map the L-bit ID into another M-bit cyclic unique codeword. An M-bit codeword $\big(w(0), w(1), \ldots, w(M-1)\big)$ is *cyclically unique* if, for any cyclic shift d, the codeword $\big(w'(0), w'(1), \ldots, w'(M-1)\big)$ is not identical to $\big(w(d), w(d+1), \ldots, w((M-1+d) \bmod M)\big)$. Such a mapping is called the *symmetrization* mapping in [26]. Then each user constructs two sequences from the deterministic modular clock algorithm: the 0-sequence with the period p_0 and the 1-sequence with the period p_1. The slopes (resp. biases) of both 0/1-sequences are set to 1 (resp. 0). The final CH sequence of a user is constructed by interleaving M 0/1-sequences according to the binary value of its M-bit cyclic unique codeword. Specifically, let $\alpha_0(t)$ (resp. $\alpha_1(t)$) be the 0-sequence (resp. 1-sequence) at time t. If $w(\tau) = 0$ (resp. 1) for $0 \leq \tau \leq M-1$, then we set the CH sequence $X(\tau + qM) = \alpha_0(q)$ (resp. $X(\tau + qM) = \alpha_1(q)$) for $q = 0, 1, 2, \ldots$. As long as the period of any 0-sequence is relatively prime to the period of any 1-sequence, the result in Corollary 3 guarantees the rendezvous of any two users within $M p_{0,\max} p_{1,\max}$ time slots, where $p_{0,\max}$ (resp. $p_{1,\max}$) is the maximum of the periods of the 0-sequences (resp. 1-sequences). One simple choice is to set the period $p_{i,0}$ of the 0-sequence of user i to be a power of 2 that is not smaller than n_i, i.e.,

$$p_{i,0} = 2^{\lceil \log_2 n_i \rceil}, \tag{3}$$

(where $\lceil x \rceil$ is the ceiling function that represents the smallest integer that is not less than x), and the period $p_{i,1}$ of the 1-sequence of user i to an odd number that is not smaller than n_i.

Clearly, for such a choice, $p_{i,0}$ and $p_{j,1}$ are relatively prime. Moreover, for all $i = 1, 2, \ldots, K$,

$$n_i \leq p_{i,1} \leq n_i + 1, \tag{4}$$

$$n_i \leq p_{i,0} < 2n_i. \tag{5}$$

From (4) and (5), any two users will rendezvous within $2Mn_{\max}(n_{\max}+1)$ time slots, where $n_{\max} = \max_{1\leq i\leq K} n_i$ is the maximum number of available channels. However, there are still two shortcomings of the above approach:

(i) The mapping from the L-bit unique ID into another M-bit cyclic unique codeword in the literature is not easy to implement if we would like to keep M close to L [26]. For instance, for a 48-bit MAC address, it was proposed in [14] that adding another 48 bits of 1's and 48 bits of 0's to the MAC address results in a 144-bit cyclically unique codeword. A mapping algorithm that requires $M = L + \lceil\sqrt{L}\rceil(2 + \lceil\log_2 L\rceil) + 3$ was proposed in [26]. For $M = 48$, this requires L to be 107.

(ii) Even though the MCTTR is bounded, the ETTR is rather poor in comparison with the random algorithm in [9]. This is because $p_{i,0}$ and $p_{j,0}$ in (3) are not relatively prime to each other and a lot of time slots are wasted when both users are using their 0-sequences.

To address these two problems, we need a stronger property than the cyclic unique property for the symmetrization class in [26].

Definition 4: **(Strong symmetrization mapping)** A set of M-bit codewords $\{\mathbf{w}_i = (w_i(0), w_i(1), \ldots, w_i(M-1)), i = 1, 2, \ldots, K\}$ is called a *strong M-symmetrization class* if for any integer d and $i \neq j$, (at least) one of the following two properties is satisfied:

(i) There exist $0 \leq \tau_1, \tau_2 \leq M-1$ such that $w_i(\tau_1) = 1, w_j((\tau_1 + d) \mod M) = 0$ and $w_i(\tau_2) = 0, w_j((\tau_2 + d) \mod M) = 1$.

(ii) There exist $0 \leq \tau_1, \tau_2 \leq M-1$ such that $w_i(\tau_1) = w_j((\tau_1 + d) \mod M) = 0$, and $w_i(\tau_2) \neq w_j((\tau_2 + d) \mod M)$.

A one-to-one mapping from the set of L-bit unique IDs to a strong M-symmetrization class is called a *strong M-symmetrization mapping*.

Clearly, a strong M-symmetrization class is an M-symmetrization class in [26]. To construct a *strong M-symmetrization mapping*, we use the $\mathcal{C}$-transform in [34], [35].

Definition 5: **($\mathcal{C}$-transform [34])** Consider an M-vector $\mathcal{U}_M = (u_1, u_2, \ldots, u_{M-1}, u_M)$ with $u_i \in \mathbf{N}$, $i = 1, 2, \ldots, M$. Define a mapping $\mathcal{C} : x \in \{0\} \cup \mathbf{N} \mapsto \{0,1\}^M$ as follows:

$$\mathcal{C}(x) = \Big(\gamma_1(x), \gamma_2(x), \ldots, \gamma_{M-1}(x), \gamma_M(x)\Big), \tag{6}$$

where

$$\gamma_M(x) = \begin{cases} 1 & \text{if } x \geq u_M \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

and for $i = M-1, \ldots, 2, 1$, $\gamma_i(x)$ is given recursively by

$$\gamma_i(x) = \begin{cases} 1 & \text{if } x - \sum_{k=i+1}^{M} \gamma_k(x) \cdot u_k \geq u_i \\ 0 & \text{otherwise} \end{cases}. \tag{8}$$

The mapping $\mathcal{C}(x)$ is called the $\mathcal{C}$-transform of x with respect to the *basis vector* $\mathcal{U}_M$. Intuitively, the $\mathcal{C}$-transform can be viewed as a "greedy" binary numeral system as the $\mathcal{C}$-transform of x is obtained by recursively subtracting x from u_M. In particular, if one selects $u_i = 2^{i-1}$ for all i, then the $\mathcal{C}$-transform of x is simply the usual binary representation of x. In the following proposition, we state two important properties of the $\mathcal{C}$-transform: the complete decomposition property (Lemma 5 of [34]) and the no consecutive 1's property (Lemma 3 of [35]).

Proposition 6:

(i) **(Complete decomposition [34])** Assume that $u_1 = 1$, and $1 \leq u_{i+1} \leq \sum_{k=1}^{i} u_k + 1, i = 1, 2, \ldots, M-1$. Then $x = \sum_{k=1}^{M} \gamma_k(x) \cdot u_k$ for $0 \leq x \leq \sum_{k=1}^{M} u_k$.

(ii) **(No ℓ consecutive 1's [35])** If, furthermore, for some $\ell \geq 2$, $u_{i+1} \leq \sum_{k=i-\ell+1}^{i} u_k$ for $i = \ell, \ldots, M-1$, then for all $0 \leq x < \sum_{k=M-\ell+1}^{M} u_k$, there are no ℓ consecutive 1's in $\mathcal{C}(x)$, i.e., there does not exist any i such that $\gamma_i(x) = \gamma_{i-1}(x) = \cdots = \gamma_{i-\ell+1}(x) = 1$.

To understand these two properties, consider the Fibonacci numeral system with the 6-vector $\mathcal{U}_6 = (1, 2, 3, 5, 8, 13)$ as the basis vector. Then for $0 \leq x \leq 32$, the $\mathcal{C}$-transform uniquely maps x to a binary 6-vector. Moreover, for $0 \leq x \leq 20$, there is no two consecutive 1's in $\mathcal{C}(x)$. For example, $\mathcal{C}(20) = (0, 1, 0, 1, 0, 1)$ and $\mathcal{C}(15) = (0, 1, 0, 0, 0, 1)$. From this example, we can map any four bit ID to a six bit codeword that does not have 2 consecutive 1's. In general, if we choose $u_i = 2^{i-1}$, $i = 1, 2, \ldots, \ell$ and $u_{i+1} = \sum_{k=i-\ell+1}^{i} u_k$ for $i = \ell, \ldots, \tilde{M}-1$ for some $\tilde{M} > \ell$, then the two properties in Proposition 6 are satisfied and thus any $\tilde{M}$-bit codeword from the $\mathcal{C}$-transform does not have ℓ consecutive 1's. Now we construct the $(\ell+2)$-bit delimiter with ℓ consecutive 1's in the middle and two 0's on both ends. We show in Theorem 7 that adding this delimiter to an $\tilde{M}$-bit codeword from the $\mathcal{C}$-transform is a strong M-symmetrization mapping with $M = \tilde{M} + \ell + 2$. Such a strong M-symmetrization mapping is outlined in Algorithm 2.

Theorem 7: Under the unique ID assumption in (A2), Algorithm 2 is a strong symmetrization mapping from an L-bit ID $\big(\gamma_1, \gamma_2, \ldots, \gamma_L\big)$ to an M-bit codeword $\big(w(0), w(1), \ldots, w(M-1)\big)$.

Proof. From Proposition 6 (ii) and Algorithm 2, we know that the substring of ℓ consecutive 1's only appears in the $(\ell+2)$-bit delimiter and thus it appears exactly once in the M-bit cyclically shifted codeword $\big(w(d), w(d+1), \ldots, w((M-1+d) \mod M)\big)$ for any integer $0 \leq d \leq M-1$. Now consider the codeword $\big(w_i(0), w_i(1), \ldots, w_i(M-1)\big)$ and the cyclically shifted codeword $\big(w_j(d), w_j(d+1), \ldots, w_j((M-1+d) \mod M)\big)$.

Case 1. $(d \mod M) \neq 0$:

In this case, the $(\ell+2)$-bit delimiters of two M-bit codewords are not aligned. Then we have $w_i(1) = \ldots = w_i(\ell) = 1$ and $w_j(t + d) \mod M))$, $t = 1, \ldots, \ell$, cannot be all 1. Thus, there exists $1 \leq \tau_1 \leq \ell$ such that $w_i(\tau_1) = 1$ and $w_j((\tau_1 + d) \mod M) = 0$. On the other hand, we have $w_j(1) = \ldots = w_j(\ell) = 1$ and $w_i((t-d) \mod M))$, $t = 1, \ldots, \ell$, cannot

Algorithm 2 The $\mathcal{C}$-transform strong symmetrization mapping

Require: An L-bit unique ID $\big(\gamma_1, \gamma_2, \ldots, \gamma_L\big)$ and a parameter ℓ.

Ensure: An M-bit codeword $\big(w(0), w(1), \ldots, w(M-1)\big)$ in a strong M-symmetrization class.

1: Construct the basis vector by letting $u_i = 2^{i-1}$, $i = 1, 2, \ldots, \ell$ and $u_{i+1} = \sum_{k=i-\ell+1}^{i} u_k$ for $i = \ell, \ldots, \tilde{M} - 1$, where $\tilde{M}$ is the smallest integer such that $\sum_{k=\tilde{M}-\ell+1}^{\tilde{M}} u_k \geq 2^L$.

2: Convert the L-bit ID into an integer $x \in [0, 2^L - 1]$ by letting $x = \sum_{i=1}^{L} \gamma_i \cdot 2^{i-1}$.

3: Use the $\mathcal{C}$-transform to compute the $\tilde{M}$-bit codeword $\mathcal{C}(x) = \big(\gamma_1(x), \gamma_2(x), \ldots, \gamma_{\tilde{M}}(x)\big)$.

4: Generate an ℓ-bit sequence with ℓ consecutive 1's and then add a 0 to each end of the ℓ-bit sequence to form an $(\ell+2)$-bit delimiter, e.g., the 6-bit delimiter 011110 for $\ell = 4$.

5: Add the $(\ell+2)$-bit delimiter in front of the $\tilde{M}$-bit codeword to form the M-bit codeword $\big(w(0), w(1), \ldots, w(M-1)\big)$, where $M = \tilde{M} + \ell + 2$, $w(0) = 0$, $w(1) = \ldots = w(\ell) = 1$, $w(\ell+1) = 0$, $w(\ell+1+i) = \gamma_i(x)$, $i = 1, 2, \ldots, \tilde{M}$.

be all 1. Thus, there exists $1 \leq ((\tau_2 + d) \bmod M) \leq \ell$ such that $w_i(\tau_2) = 0$ and $w_j((\tau_2 + d) \bmod M) = 1$.

Case 2. $(d \bmod M) = 0$: In this case, the $(\ell+2)$-bit delimiters of two M-bit codewords are aligned. Choose $\tau_1 = 0$ and we have $w_i(\tau_1) = w_j(\tau_1) = w_j((\tau_1 + d) \bmod M) = 0$. From the uniqueness of the L-bit ID assumption in (A2) and the one-to-one mapping of the $\mathcal{C}$-transform, we know there exists $\ell + 2 \leq \tau_2 \leq M - 1$ such that $w_i(\tau_2) \neq w_j(\tau_2) = w_j((\tau_2 + d) \bmod M)$. ■

The parameter ℓ in Algorithm 2 plays an important role in the code rate L/M. Note that u_k in Algorithm 2 grows asymptotically at the exponentially rate ρ, i.e., $u_k \approx \rho^k$, where ρ is the unique solution in the interval $(1, 2)$ of the following equation:

$$1 + \rho + \rho^2 + \ldots + \rho^{\ell-1} = \rho^{\ell}.$$

In particular, for $\ell = 2$, we have $\rho = \frac{1+\sqrt{5}}{2}$ and the asymptotic code rate is $\log_2 \rho \approx 0.6942$. For $\ell = 5$, we have $\rho \approx 1.965848$ and the asymptotic code rate is $\log_2 \rho \approx 0.9752$. Thus, M grows linearly in L in the asymptotic regime and the asymptotic code rate is very close 1 even for a small ℓ. The computational complexity of the $\mathcal{C}$-transform strong symmetrization mapping for an L-bit ID is $O(M)$ and thus $O(L)$. However, as shown in [35], the hardware implementation complexity (in terms of the number of logic gates) is $O(M^2)$ and that might pose a serious hardware design challenge even for a 48-bit MAC address.

One trick to reduce the hardware implementation complexity of the $\mathcal{C}$-transform strong symmetrization mapping is to divide the L-bit ID into k parts with

$$L = L_1 + L_2 + \ldots L_k.$$

TABLE II: The 4B5B encoding table

4B data	5B code	4B data	5B code
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

Then use the $\mathcal{C}$-transform with the parameter ℓ for each part to construct an $\tilde{M}_k$-bit codeword. By inserting a 0 between the two codewords of two successive parts, we then have an $\tilde{M}$-bit codeword that does not have ℓ consecutive 1's, where $\tilde{M} = \tilde{M}_1 + \ldots \tilde{M}_k + k - 1$. As in Algorithm 2, the last step is to add the $(\ell+2)$-bit delimiter to the $\tilde{M}$-bit codeword to form an M-bit codeword with $M = \tilde{M} + \ell + 2$. For example, suppose $L = 48$ (for a MAC address). We divide it into two parts, each with 24 bits. For $\ell = 5$, we then have $\tilde{M}_1 = \tilde{M}_2 = 25$, $\tilde{M} = 51$, and $M = 58$. This is much smaller than 144 in [14] and 107 in [26].

C. 4B5B encoding

In this section, we propose a much simpler strong symmetrization mapping than the $\mathcal{C}$-transform in the previous section. The idea is to use the standard 4B5B coding of the L-bit unique ID to construct a strong M-symmetrization mapping, where $M = \lceil L/4 \rceil * 5 + 6$. In particular, for $L = 48$, we have $M = 66$, which is slightly larger than 58 by using the $\mathcal{C}$-transform at the end of the previous section. The 4B5B encoding scheme is widely used in computer networks (see e.g., [36]). In such an encoding scheme, each piece of 4 bits is uniquely mapped to a 5-bit codeword (see Table II). One salient feature of the 4B5B encoding scheme is that each 5-bit codeword has at most one leading 0 as well as at most two trailing 0's. Thus, encoding the L-bit ID results in a $\lceil L/4 \rceil * 5$-bit codeword that does not have 4 consecutive 0's. Now we add the 6-bit delimiter 100001 in front of the $\lceil L/4 \rceil * 5$-bit codeword to construct an $M = \lceil L/4 \rceil * 5 + 6$ codeword. The details of the mapping from an L-bit ID to an M-bit codeword is shown in Algorithm 3.

Algorithm 3 The 4B5B strong symmetrization mapping

Require: An L-bit unique ID.

Ensure: An M-bit cyclic unique codeword $\big(w(0), w(1), \ldots, w(M-1)\big)$, where $M = \lceil L/4 \rceil * 5 + 6$.

1: If L is not an integer multiple of 4, append $4 - (L \bmod 4)$ 0's to the unique ID to form a $\lceil L/4 \rceil * 4$-bit ID.

2: Use the 4B5B encoding scheme to encode the $\lceil L/4 \rceil * 4$-bit ID into a $\lceil L/4 \rceil * 5$-bit codeword.

3: Add the 6-bit delimiter 100001 in front of the $\lceil L/4 \rceil * 5$-bit codeword to form a $(\lceil L/4 \rceil * 5 + 6)$-bit codeword.

In the following lemma, we show that the 4B5B mapping in Algorithm 3 is indeed a strong symmetrization mapping.

Lemma 8: Under the assumption in (A2), let $\big(w_i(0), w_i(1), \ldots, w_i(M-1)\big)$ be the codeword generated

from Algorithm 3 by using the L-bit ID of user i. Then such a mapping is a strong M-symmetrization mapping.

Proof. From the 4B5B mapping in Algorithm 3, we know that the substring of 4 consecutive 0's only appears in the 6-bit delimiter 100001 and thus it appears exactly once in the M-bit codeword for any cyclic shift d. By inverting each bit of the M-bit codeword, we have the 6-bit delimiter 011110 and the substring of 4 consecutive 1's appears exactly once in the inverted M-bit codeword for any cyclic shift d. Following the same argument as that in the proof of Theorem 7 then leads to the desired result. ■

D. Two-prime modular clock algorithm

Now we combine the deterministic modular clock algorithm in Algorithm 1 and the strong symmetrization mappings in Algorithm 2 and Algorithms 3 to construct a CH that can provide guaranteed rendezvous under the assumption that each user is assigned with a unique ID. Such an algorithm is called the *two-prime modular clock algorithm* in this paper and its detail is shown in Algorithm 4. The idea, as described before, is to interleave M 0/1-sequences according to the binary value of its M-bit codeword from the strong symmetrization mapping of the L-bit ID. For user i, we selects two *primes* $p_{i,0}$ and $p_{i,1}$ such that $n_i \leq p_{i,0} < p_{i,1}$. A 0-sequence (resp. 1-sequence) of user i is then constructed by using the deterministic modular clock algorithm with the prime $p_{i,0}$ (resp. $p_{i,1}$). The slope parameter and the bias parameter are determined by a deterministic hash function so that these two parameters appear to be "random." Then the CH sequence of a user is constructed by interleaving M 0/1-sequences according to its M-bit codeword.

We note that it is possible that $p_{i,0} = p_{j,1}$ and thus the previous relatively prime argument for interleaving M 0/1-sequences according to cyclic unique codewords fails. Fortunately, the two properties for a strong symmetrization mapping is much stronger than the cyclic unique property and we can use them to prove guaranteed rendezvous in the following theorem.

Theorem 9: Suppose the assumptions in (A1) and (A2) hold and all the K users use Algorithm 4 to generate their CH sequences. Then user i and user j will rendezvous on every common available channel at least once within $M \max[p_{i,0}p_{j,1}, p_{i,1}p_{j,0}]$ time slots.

Proof. Let d be the clock shift between these two users. Note from Algorithm 4 that for $t \in \{\tau, \tau + M, \tau + 2M, \ldots\}$, user i uses a $w_i(\tau)$-sequence and user j uses a $w_j(\tau + d)$-sequence. In view of the definition of the strong M symmetrization mapping in Definition 4, we consider the following two cases.

Case 1. There exist $0 \leq \tau_1, \tau_2 \leq M - 1$ such that $w_i(\tau_1) = 1, w_j((\tau_1 + d) \bmod M) = 0$ and $w_i(\tau_2) = 0, w_j((\tau_2 + d) \bmod M) = 1$:

In this case, for $t \in \{\tau_1, \tau_1 + M, \tau_1 + 2M, \ldots\}$, user i uses a 1-sequence and user j uses a 0-sequence. The 1-sequence of user i is generated from the deterministic modular clock algorithm with the prime $p_{i,1}$ and and the 0-sequence of user j is generated from the deterministic modular clock algorithm with the prime $p_{j,0}$. If $p_{i,1} \neq p_{j,0}$, then we conclude from Corollary 3 that these two users will rendezvous on every common available channel at least once within $Mp_{i,1}p_{j,0}$ time slots.

On the other hand, if $p_{i,1} = p_{j,0}$, then we have

$$p_{j,1} > p_{j,0} = p_{i,1} > p_{i,0}.$$

Now for $t \in \{\tau_2, \tau_2 + M, \tau_2 + 2M, \ldots\}$, user i uses a 0-sequence and user j uses a 1-sequence. The 0-sequence of user i is generated from the deterministic modular clock algorithm with the prime $p_{i,0}$ and and the 1-sequence of user j is generated from the deterministic modular clock algorithm with the prime $p_{j,1}$. Since $p_{j,1} \neq p_{i,0}$, we know from Corollary 3 that these two users will rendezvous on every common available channel at least once within $Mp_{i,0}p_{j,1}$ time slots.

Case 2. There exist $0 \leq \tau_1, \tau_2 \leq M - 1$ such that $w_i(\tau_1) = w_j((\tau_1 + d) \bmod M) = 0$, and $w_i(\tau_2) \neq w_j((\tau_2 + d) \bmod M)$:

In this case, for $t \in \{\tau_1, \tau_1 + M, \tau_1 + 2M, \ldots\}$, user i uses a 0-sequence and user j uses a 0-sequence. The 0-sequence of user i is generated from the deterministic modular clock algorithm with the prime $p_{i,0}$ and the 0-sequence of user j is generated from the deterministic modular clock algorithm with the prime $p_{j,0}$. If $p_{i,0} \neq p_{j,0}$, then we conclude from Corollary 3 that these two users will rendezvous on every common available channel at least once within $Mp_{i,0}p_{j,0}$ time slots.

On the other hand, if $p_{i,0} = p_{j,0}$, then we have

$$\begin{aligned} p_{i,1} &> p_{i,0} = p_{j,0}, \\ p_{j,1} &> p_{j,0} = p_{i,0}. \end{aligned} \tag{9}$$

Now for $t \in \{\tau_2, \tau_2 + M, \tau_2 + 2M, \ldots\}$, user i uses a $w_i(\tau_2)$-sequence and user j uses a $w_j(\tau_2 + d)$-sequence with $w_i(\tau_2) \neq w_j(\tau_2 + d)$. In view of (9), we conclude from Corollary 3 that these two users will rendezvous on every common available channel at least once within $M \max[p_{i,0}p_{j,1}, p_{i,1}p_{j,0}]$ time slots. ■

Since there is a prime between $[n, 2n]$ [37] and another prime in $[2n, 3n]$ [38], we then have the following corollary.

Corollary 10: Suppose the assumptions in (A1) and (A2) hold and all the K users use Algorithm 4 to generate their CH sequences. Furthermore, user i chooses $p_{i,0}$ as the smallest prime not smaller than n_i and $p_{i,1}$ as the smallest prime larger than $p_{i,0}$, $i = 1, 2, \ldots, K$. Then user i and user j will rendezvous on every common available channel at least once within $6Mn_in_j$ time slots. In particular, for the 4B5B strong symmetrization mapping, we have $M = (\lceil L/4 \rceil * 5 + 6)$ for an L-bit ID.

Now we comment on the ETTR of the two-prime modular clock algorithm. As mentioned at the beginning of this section, one way to reduce the ETTR is to avoid introducing "stay" modes that repeatedly examine the same channel pairs of two users. As such, the slope r chosen in Line 8 of our algorithm is an integer in $[1, p-1]$ and it changes in every time slot. As the slope r is nonzero, there is no "stay" mode in our algorithm. On the other hand, the bias b chosen in Line 9 of our algorithm is an integer in $[0, p-1]$. Thus, the total number of "lines"

Algorithm 4 The two-prime modular clock algorithm

Require: An available channel set $\mathbf{c} = \{c(0), c(1), \ldots, c(n-1)\}$, two primes $p_1 > p_0 \geq n$, and an L-bit ID.

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

1: Use a strong M-symmetrization mapping (such as Algorithm 2 or Algorithm 3) to construct an M-bit codeword $(w(0), w(1), \ldots, w(M-1))$ from the L-bit ID.
2: Let $z = 0$.
3: For each t, compute the following variables:
4: $q = \lfloor t/M \rfloor$.
5: $s = (t \bmod M)$.
6: $p = p_{w(s)}$.
7: $y = (s \bmod (p(p-1)))$.
8: $r = (y \bmod (p-1)) + 1$.
9: $b = \lfloor y/(p-1) \rfloor$.
10: $k = ((r * q + b) \bmod p)$.
11: If $k \leq n-1$, let $X(t) = c(k)$.
12: Otherwise, let $X(t) = c(z)$ and update $z \leftarrow ((z+1) \bmod n)$.

with nonzero slopes is $p(p-1)$ and each y in Line 7 of our algorithm corresponds to one of the time-interleaved $p(p-1)$ "lines". As such, our algorithm selects each available channel with an equal probability in the long run and thus appears to be random. In Section IV, we will show by computer simulations that the ETTR of our algorithm for two-user rendezvous is almost the same as that of the random algorithm.

To provide a rigorous argument for the ETTR of the two-prime modular clock algorithm, let us consider using (deterministic) pseudorandom number generators to generate the slope r and the bias b (instead of using the deterministic hash functions in Algorithm 4). Specifically, let $h_1(t,p)$, $h_2(t,p)$ and $h_3(t,p)$ be three pseudorandom number generators that return "independent" and "uniformly distributed" integers in $[0, p-1]$ with the seed t. Then we can replace $r = h_1(s, p-1) + 1$ in Line 8, $b = h_2(s,p)$ in Line 9 and $X(t) = c(h_3(t,n))$ in Line 12. If the same p is used for $t = 0, 1, 2, \ldots, M-1$, then the integers $k(t)$ in Line 10 are clearly independent and uniformly distributed in $[0, p-1]$. Even though we use two different primes p_0 and p_1, the family of random variables $\{X(t), t = 0, 1, 2, \ldots, M-1\}$, in Lines 11 and 12 are independent as they are deterministic functions of the three independent random variables, $h_1(s, p-1)$, $h_2(s,p)$ and $h_3(t,n)$. It is also not hard to see that $X(t)$, $t = 0, 1, 2, \ldots, M-1$, are chosen from the available channel set $\mathbf{c} = \{c(0), c(1), \ldots, c(n-1)\}$ with equal probabilities. Thus, for $t = 0, 1, 2, \ldots, M-1$, the two-prime modular clock algorithm behaves as if it were the random algorithm. On the other hand, we note that $X(t)$ and $X(t+qM)$ are not independent as they both have the same s and thus the same r and b. Such a correlated property ensures that the MCTTR is bounded as proved in Theorem 9.

Now we use these two properties to bound the ETTR of the two-prime modular clock algorithm. Let $h = n_{i,j}/n_i n_j$ be the probability that the two users hop on one common available channel by using the random algorithm. Clearly, the ETTR of the random algorithm is $1/h$. Also, let $H = M \max[p_{i,0}p_{j,1}, p_{i,1}p_{j,0}]$ be the upper bound for the time-to-rendezvous T in Theorem 9. As the two-prime modular clock algorithm behaves as if it were the random algorithm for the first M time slots, we have

$$\begin{aligned}
\mathsf{E}[T] &= \sum_{t=1}^{H} t \cdot \mathsf{P}(T=t) \\
&= \sum_{t=1}^{M} t \cdot \mathsf{P}(T=t) + \sum_{t=M+1}^{H} t \cdot \mathsf{P}(T=t) \\
&= \sum_{t=1}^{M} t \cdot h(1-h)^{t-1} + \sum_{t=M+1}^{H} t \cdot \mathsf{P}(T=t) \\
&\leq \sum_{t=1}^{\infty} t \cdot h(1-h)^{t-1} + H \sum_{t=M+1}^{H} \mathsf{P}(T=t) \\
&= 1/h + H \cdot \mathsf{P}(T > M) \\
&= 1/h + H \cdot (1-h)^M. \qquad (10)
\end{aligned}$$

As H is linear in M, the second term converges to 0 as $M \to \infty$. Thus, the ETTR of the two-prime modular clock algorithm is almost the same as that of the random algorithm when M is large.

The idea of using two primes to generate CH sequences is not new. For instance, under the two-radio assumption, i.e., each user can hop on two channels at the same time, it was shown in Algorithm 2 of [23] that the two users can rendezvous in $O(n_i n_j)$ time slots. By using the strong symmetrization mapping, our two-prime modular clock algorithm does this with a single radio.

As an illustrating example, let us consider a CRN with two users and five channels $\{0, 1, 2, 3, 4\}$. The available channel set for user 1 (resp. user 2) is $\mathbf{c}_1 = \{c_1(0), c_1(1), c_1(2)\} = \{0, 2, 4\}$ (resp. $\mathbf{c}_2 = \{c_2(0), c_2(1), c_2(2)\} = \{3, 0, 1\}$). Thus, both users have three available channels and the only common available channel is channel 0. Each user is assigned with a four-bit ID, i.e., $L = 4$, and the ID of user 1 (resp. user 2) is 0100 (resp. 0001). Using the 4B5B mapping in Algorithm 3 yields a 10-bit codeword 10000101010 for user 1 (resp. 10000101001 for user 2). Thus, $M = 10$. As each user has three available channels, we have $p_{1,0} = p_{2,0} = 3$ and $p_{1,1} = p_{2,1} = 5$ for the two-prime modular clock algorithm. In Figure 1, we show the CH sequences for these two users from $t = 0$ (the top sequence for user 1 and the bottom sequence for user 2). Note that the difference between the two local clocks of these two users is 3 in this figure.

III. Multiuser rendezvous

For the multiuser rendezvous problem, one commonly used approach is to achieve multiuser rendezvous by a series of pair-wise rendezvous (see e.g., [11], [13], [19]). The basic idea, as described in [11], [13], [19], is to elect a leader when a set of users rendezvous on a channel. Then every user in this set hops along with the leader. To work with the two-prime modular clock algorithm in Algorithm 4, each user is required to maintain the following state information:

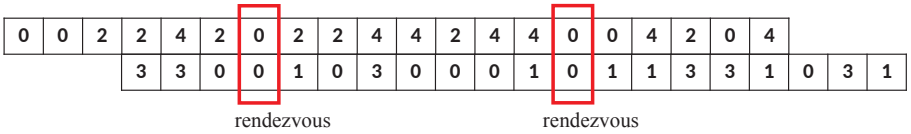

Fig. 1: An illustrating example for the two-prime modular clock algorithm with two users (the top sequence for user 1 and the bottom sequence for user 2).

(i) user's available channel set

$$\mathbf{c} = \{c(0), c(1), \ldots, c(n-1)\},$$

(ii) leader's available channel set

$$\mathbf{c}' = \{c'(0), c'(1), \ldots, c(n'-1)\},$$

(iii) the common channel set

$$\mathbf{c}'' = \{c''(0), c''(1), \ldots, c(n''-1)\},$$

(iv) leader's two primes $p_1 > p_0 \geq n'$, and

(v) leader's L-bit ID.

Initially, each user considers itself as a leader and thus sets leader's available channel set and the common channel set to be the same as its available channel set. Then each user selects two primes following the selection in Corollary 10 and sets leader's L-bit ID according to its own ID. When a set of users rendezvous on a channel, they exchange their *state information* and their *local clocks*. Then a leader is elected among this set of users. Each user in this set computes the new common channel set from the intersection of the common channel sets of these users. Moreover, each user in this set synchronizes its clock to the clock of the leader and update its leader's available channel set, leader's two primes and leader's L-bit ID to generate its CH sequence onwards. For instance, if user i and user j rendezvous on a channel and user i is elected as the leader, then user i (as the leader) updates its "common channel set" by taking the intersection of $\mathbf{c}_i''$ and $\mathbf{c}_j''$. Similarly, user j also updates its "common channel set" by taking the intersection of $\mathbf{c}_i''$ and $\mathbf{c}_j''$. Moreover, user j synchronizes its clock to user i's clock and replaces its "leader's available channel set", "leader's two primes", and "leader's L-bit ID" by user i's "leader's available channel set", "leader's two primes", and "leader's L-bit ID".

There are two issues that need to be further clarified: (i) the method of electing the leader, and (ii) the method of replacing a channel in the CH sequence of the leader that is not an available channel to a user following the leader. Since we assume that each user is assigned with a unique ID, one common way to address the problem of electing a leader is to elect the user with the largest ID. But this may not be a good choice for the deterministic modular clock algorithm as the period of the user with the largest ID might be very large. A better alternative is to select the leader with the smallest number of available channels. If there is a tie, then the user with the largest ID among the users with the smallest number of available channels is elected. On the other hand, one solution for the second problem is for each user to hop only on the channels in the common channel set. This is outlined in the stick together algorithm in Algorithm 5.

Algorithm 5 The stick together channel hopping algorithm

Require: User's available channel set $\mathbf{c} = \{c(0), c(1), \ldots, c(n-1)\}$, leader's available channel set $\mathbf{c}' = \{c'(0), c'(1), \ldots, c(n'-1)\}$, a common channel set $\mathbf{c}'' = \{c''(0), c''(1), \ldots, c(n''-1)\}$, leader's two primes $p_1 > p_0 \geq n'$, and leader's L-bit ID.

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

1: Use a strong M-symmetrization mapping (such as Algorithm 2 or Algorithm 3) to construct an M-bit codeword $(w(0), w(1), \ldots, w(M-1))$ from leader's L-bit ID.
2: Let $z = 0$.
3: For each t, follow Algorithm 4 to compute the clock k.
4: If $c'(k)$ is a channel in the common channel set, i.e., $c'(k) \in \mathbf{c}''$, let $X(t) = c'(k)$.
5: Otherwise, select a channel from the *common* channel set by letting $X(t) = c''(z)$ and update $z \leftarrow ((z+1) \bmod n'')$.

Since Algorithm 5 is a deterministic algorithm, the set of users then hops along with the leader after their rendezvous. This is why we call Algorithm 5 the stick together algorithm. In the following theorem, we show an upper bound for the MCTTR of K users.

Theorem 11: Suppose the assumptions in (A1) and (A2) hold and each user chooses its initial two primes as in Corollary 10 and uses Algorithm 5 to generate its CH sequence. When a set of users rendezvous on a channel, the user with the largest ID among the user(s) with the smallest number of available channels in this set of users is elected as the leader of this set of users. Then all the K users will rendezvous on a common channel in $6Mn_{\min}n_{\max}$ time slots, where

$$n_{\min} = \min_{1 \leq i \leq K} n_i, \tag{11}$$

$$n_{\max} = \max_{1 \leq i \leq K} n_i, \tag{12}$$

are the minimum number and the maximum number of available channels among the K users, respectively.

Proof. Let $\mathbf{c}_{\text{all}} = \cap_{i=1}^{K} \mathbf{c}_i$ be the set of common channels. The key insight of Algorithm 5 is that it is a deterministic algorithm and the time slots that a user hops on a channel in $\mathbf{c}_{\text{all}}$ will not be changed as long as it is still a leader. Once a leader becomes a follower of a leader after a group rendezvous, the follower will hop on a channel in $\mathbf{c}_{\text{all}}$ at the same time slots as its leader. From the way we select a leader, there is a total ordering of these K users. Suppose that user i is the user with the largest ID among the user(s) with the smallest number of available channels. Thus, $n_i = n_{\min}$ and user i is always elected as the leader every time it makes a rendezvous with other users on a channel. As such, the parameters needed in Algorithm 5 for user i is never changed except the common channel set is getting smaller after a rendezvous and eventually converges to $\mathbf{c}_{\text{all}}$. Thus, the time slots that user i hops on a channel in $\mathbf{c}_{\text{all}}$ are never changed. Call these time slots of user i as rendezvous time slots. From Corollary 10, we know that user i will rendezvous with other leaders (and their followers) on these rendezvous time slots within $6Mn_{\min}n_{\max}$ time slots. ■

One interesting question is whether the stick together policy is a good policy. It was argued and shown by computer simulation in [13] that it might be helpful for users to spread out to increase the opportunity to find other users. In order to have the guaranteed rendezvous property in Theorem 11, the spread out policy has to be implemented with care. For this, we modify Algorithm 5 by allowing each user to use both its available channel set and the common channel set to generate its CH sequence. This is outlined in Algorithm 6.

Algorithm 6 The spread out channel hopping algorithm

Require: User's available channel set $\mathbf{c} = \{c(0), c(1), \ldots, c(n-1)\}$, leader's available channel set $\mathbf{c}' = \{c'(0), c'(1), \ldots, c(n'-1)\}$, a common channel set $\mathbf{c}'' = \{c''(0), c''(1), \ldots, c(n''-1)\}$, leader's two primes $p_1 > p_0 \geq n'$, and leader's L-bit ID.

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

1: Use a strong M-symmetrization mapping (such as Algorithm 2 or Algorithm 3) to construct an M-bit codeword $(w(0), w(1), \ldots, w(M-1))$ from the L-bit ID.
2: Let $z = 0$.
3: For each t, follow Algorithm 4 to compute the clock k.
4: If $c'(k)$ is a channel in the common channel set, i.e., $c'(k) \in \mathbf{c}''$, let $X(t) = c'(k)$.
5: Otherwise, select a channel from the *available* channel set by letting $X(t) = c(z)$ and update $z \leftarrow ((z+1) \bmod n)$.

The only difference between the stick together algorithm in Algorithm 5 and the spread out algorithm in Algorithm 6 is the last step. In the stick together (resp. spread out) algorithm, a channel is selected from the common (resp. available) channel set. As such, if $X(t)$ is a common channel, then all the users following the same leader still hop to the same channel. Such a property ensures that the worst case result in Theorem 11 still holds. In the experiment section, we will compare the ETTR for these two algorithms.

IV. Experiments

In this section, we conduct extensive simulations to compare the performance of our two-prime modular clock algorithm with that of some best-performed CH algorithms in the literature, including Modified Modular Clock Algorithm [9], FRCH [17], and Advanced Rendezvous Protocol [26].

A. Simulation settings

In our experiments, there are $N = 50$ channels, indexed from 0 to 49. In order to satisfy the assumption that there is at least one common available channel in (A1), channel 0 is chosen to be in the available channel set for each user. For the rest 49 channels, we randomly assign them to the available channel set of each user. Specifically, for user i, $i = 1, 2, \ldots, K$, we assign a system parameter, called the *channel availability probability* v_i. Each of the 49 remaining channels is assigned *independently* to the available channel set of user i with probability v_i. In Figure 2, we show the expected number of common available channels among K users as a function of the channel availability probability when $v_i = v$ for all $i = 1, 2, \ldots, K$. Such curves are generated by averaging over 100,000 simulations. It is clear to see that the expected number of common available channels is increasing in the channel availability probability, ranging from the lowest value 1 (when $v = 0$) to the largest value 50 (when $v = 1$). The expected number of common available channels is also decreasing in the number of users K and it is more difficult to have more than one common available channel when K is large and v is small.

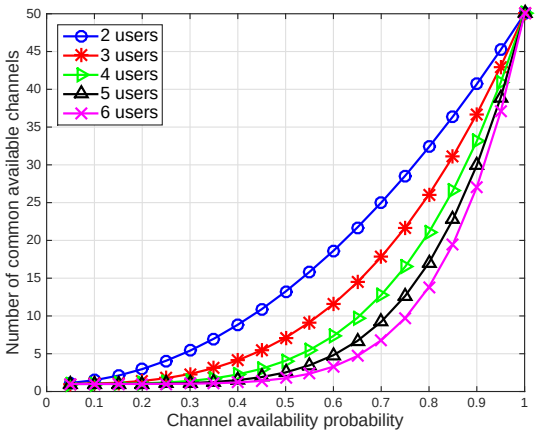

Fig. 2: The expected number of common available channels as a function of the channel availability probability and the number of users.

For the unique ID assumption in (A2), we assign each user a randomly generated 48-bit ID (to mimic its 48-bit MAC address). The 48-bit ID is transformed into a 66-bit codeword by the 4B5B mapping in Algorithm 3 for our two-prime modular clock algorithm. The clock drift of each user for our

two-prime modular clock algorithm is uniformly selected in $[0, T-1]$, where $T = 66 * p_0 * (p_0 - 1) * p_1 * (p_1 - 1)$. On the other hand, we also use the η_1-symmetrization mapping in [26] to transform the 48-bit ID into a 99-bit cyclically unique codeword for the Advanced Rendezvous Protocol in [26]. The clock drift of each user for the Advanced Rendezvous Protocol in [26] is uniformly selected in [0,98].

B. Two-user rendezvous

In Figure 3, we show the simulation results for the ETTRs of these CH algorithms with two users. Each data point in this figure is obtained by averaging over 100,000 simulations. The "lower bound" curve is the average of the lower bound $(n_i n_j + 1)/(n_{i,j} + 1)$ for ETTR in Proposition 1 over 100,000 simulations. As shown in Figure 3, the ETTR of our two-prime modular clock algorithm basically overlaps with that of the random algorithm and it is significantly better than the other three CH algorithms. In fact, after examining the numerical values of the simulation results, our two-prime modular clock algorithm is slightly better than the random algorithm when the channel availability probability is in the range of $[0.25, 1]$ and in that range it is nearly optimal as it is very close to the lower bound curve.

The reason that the modified modular clock algorithm in [9] does not perform well is because the channel availability probabilities are set to be the same for these two users and thus the difference of the number of available channels for these two users is small. As such, it is very likely for the modified modular clock algorithm in [9] to select the same prime for these two users at the beginning. As it takes a long time to detect the need to change the primes of the two users, a lot of time slots are wasted and that leads to a long ETTR. On the other hand, our two-prime modular clock algorithm does not have that problem as it has been solved by the strong symmetrization mapping in Theorem 9.

FRCH in [17] uses the DRSEQ algorithm [8] to generate its default sequence. When all the channels are available to both users, it was proved in [28] that a randomized version of the DRSEQ algorithm has a slightly shorter ETTR than that of the random algorithm. As such, FRCH performs roughly as well as the random algorithm in our simulations when the channel availability probability v is 1. However, the adaptive sequence of FRCH has a "stay" mode that replaces each unavailable channel in its default sequence by the same channel in its available channel set for a duration of $2N + 1$ time slots, where N is the total number of channels. As shown in Figure 3, the ETTR of FRCH is much worse than that of the random algorithm when the channel availability probability v is less than 1. We note that FRCH does not need the unique ID assumption in (A2). But it needs a universal channel labelling method for the N channels.

The Advanced Rendezvous Protocol in [26] also assumes the unique ID assumption in (A2). For an 48-bit ID, The η_1-symmetrization mapping adds a delimiter that has 49-bit of 0's in the middle and two 1's at the both ends. As such, there are many 0's in the 99-bit cyclically unique codeword and thus both users use their 0-sequences very often. Since there is no guaranteed rendezvous when both users use their 0-sequences, a lot of time slots are wasted. On the other hand, the 4B5B strong symmetrization mapping in Algorithm 3 does not yield a lot of consecutive 0's or 1's in its codeword and it is a better way to encode the ID than the η_1-symmetrization mapping in [26].

The Conversion Based Hopping (CBH) algorithm in [25] also assumes the unique ID assumption in (A2). The ID conversion is based on the $(p-1)$-ary representation of an ID, where p is a prime not smaller than the maximum of the number of available channels and 3. Obtaining the $(p-1)$-ary representation for a 48-bit integer is not as easy as the 4B5B strong symmetrization mapping in Algorithm 3. For simplicity, we randomly choose the ID of a user uniformly in [1,100] for our simulations of the CBH algorithm. As shown in Figure 3, the ETTR of CBH is larger than that of our two-prime modular clock algorithm (and the random algorithm). One possible explanation for this is that CBH uses a deterministic mapping from the p "logical" channels to the available channels (see Line 17 of Algorithm 3 in [25]) and the load is thus not evenly distributed among the available channels.

The CH sequence generating algorithm (CHGA) in [30] is a hybrid CH algorithm that interleaves the simple random algorithm with a deterministic CH algorithm (such as CRSEQ [7] and JS [11]). In our simulations, it is generated by using the wake-up sequence {1,1,1,0,1,0,0,0} and the JS algorithm [11]. When the designated output of the JS algorithm is not in the available channel set, we simply choose an available channel uniformly at random. Such a wake-up schedule corresponds to the duty cycle of 50% and thus both users of CHGA use the simple random algorithm for more than 50% of the time. As shown in Figure 3, its ETTR is still slightly larger than our two-prime modular clock algorithm when the channel availability probability v is in the range of $[0.1, 0.6]$. When the channel availability probability is in the range of $[0.6, 1]$, the ETTRs of both algorithms are almost the same and they are very close to the lower bound curve. When the duty cycle of the wake-up schedule of CHGA is increased to 100%, it reduces to the JS algorithm [11]. As shown in Figure 3, the ETTR of the JS algorithm is almost the same as that of CHGA when v is less than 0.3 (as the JS algorithm behaves as if it were a random algorithm in that range). When v is in the range of $[0.3, 0.6]$, we start to see the effect of the "stay" mode and the ETTR of the JS algorithm is slightly larger than that of CHGA. Such an effect will be more apparent in the multiuser rendezvous setting. When v is in the range of $[0.8, 1]$, the ETTR of the JS algorithm drops significantly and performs much better than the other algorithms as the JS algorithm is able to "learn" from unsuccessful rendezvous by using the universal labelling of the channels.

In Figure 4, we also compare the simulated MCTTRs for five algorithms that achieve guaranteed rendezvous: the two-prime modular clock algorithm, CBH [25], the Advanced Rendezvous Protocol in [26], CHGA in [30], JS in [11] and FRCH in [17]. The simulated MCTTR is measured by the maximum of the TTRs of 10,000 independent simulations. Each data point in Figure 4 is then obtained by averaging over 100 simulated MCTTRs. As shown in Figure 4, the simulated

MCTTR of our two-prime modular clock algorithm is still significantly better than that of the Advanced Rendezvous Protocol in [26]. This is because both the MCTTRs of our two-prime modular clock algorithm and the Advanced Rendezvous Protocol in [26] depend on the number of coded bits M. Since the 48-bit ID is used, we have $M = 66$ for our two-prime modular clock algorithm and $M = 99$ for the Advanced Rendezvous Protocol in [26]. The simulated MCTTR of CBH [25] is slightly larger than that of the Advanced Rendezvous Protocol in [26] (even the IDs in CBH are chosen in [1,100]). This might be because the MCTTR of CBH is $O(\max[n_i, n_j]^2)$ and that of the Advanced Rendezvous Protocol is only $O(n_i \cdot n_j)$. The simulated MCTTR of CHGA in [30] is comparable with that of the two-prime modular clock algorithm. On the other hand, the simulated FRCH could be very large when the channel availability probability is slightly less than 1. But when all the channels are available, both the simulated MCTTRs of FRCH and JS drop significantly due to their ability to learn from unsuccessful rendezvous.

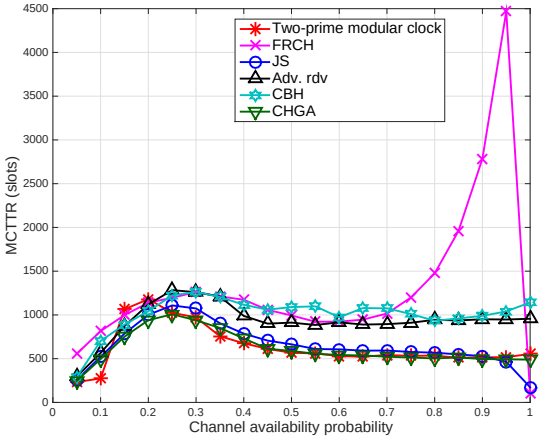

Fig. 4: Comparison of the MCTTRs of various CH algorithms, including FRCH [17], CBH [25], the advanced rendezvous protocol (Adv. rdv) [26], CHGA [30], JS [11] and our two-prime modular clock algorithm.

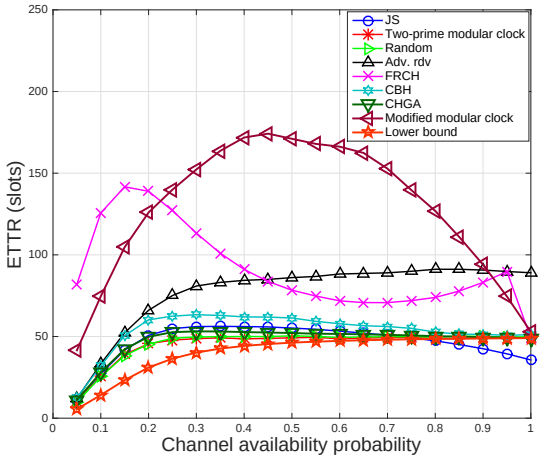

Fig. 3: Comparison of the ETTRs of various CH algorithms, including the random algorithm, the modified modular clock algorithm [9], FRCH [17], CBH [25], the advanced rendezvous protocol (Adv. rdv) [26], CHGA [30], JS [11] and our two-prime modular clock algorithm. The "lower bound" curve is the average of the lower bound $(n_i n_j+1)/(n_{i,j}+1)$ for ETTR in Proposition 1 over 100,000 simulations.

C. Multiuser rendezvous

To evaluate the performance of the multiuser rendezvous algorithms, we compare the simulation results from our algorithms with those from the random algorithm. To extend the random algorithm for multiuser rendezvous, each user considers itself as the leader at the beginning and it also keeps the information of the common channel set. As in the stick together algorithm, when a group of users rendezvous on a channel in the random algorithm, they elect a leader, compute the new common channel set, and then hop together with the leader. At every time slot, each leader uniformly selects a channel in its common channel set. In Figure 5, we show the simulation results for three users with the same channel availability probability v. Each data point in the graph is also obtained by averaging over 100,000 simulations. To our surprise, the spread out algorithm (Algorithm 6) is not always better than the stick together algorithm (Algorithm 5) as commonly claimed in the literature (see e.g., [13]). As shown in Figure 5, the ETTR curve of the stick together algorithm (Algorithm 5) and that of the random algorithm almost overlap with each other except only a few points that the stick together algorithm outperforms the random algorithm. The spread out algorithm (Algorithm 6) does not perform as well as the other two algorithms when the channel availability probability v is small. On the other hand, it outperforms the other two when v is larger than 0.3. The intuition behind this is that when v is the small, the number of common channels for the three users is very small. Under such a scenario, the spread out algorithm that hops on one of its available channels does not improve the rendezvous probability. As such, it might be better to simply hop on one of the common channels as in the stick together algorithm. In particular, when there is only one common channel and that channel is known to a user, the optimal policy for that user is to hop on that common channel all the time. On the other hand, if there are lots of common channels between any pair of two users, then the spread out algorithm does improve the rendezvous probability. When a group of users rendezvous, the state information can be "synchronized" among this group of users and users with "synchronized" state information (including the leader ID and the common channel set) tend to rendezvous faster. Such a phenomenon is similar to that in the distributed consensus problem [39], [40].

The random algorithm and our algorithms are targeted for the oblivious rendezvous setting, where there are no universal labelling of the channels. To see the effect of having a universal labelling of the channels, we also conduct the simulation for the Jump-stay (JS) algorithm in [11] for multiuser

rendezvous. We note that the JS algorithm requires a universal labelling of the channels and it cannot be applied to the oblivious rendezvous problem. As shown in Figure 5, the ETTR of the JS algorithm for $K = 3$ is significantly higher than our algorithms when v is small. This is mainly due to the effect of the long "stay" mode in the JS algorithm. On the other hand, when v is close to 1, the ETTR of the JS algorithm decreases rapidly and it performs even better than the random algorithm and our algorithms. This is because when all the channels are available to every user, the benefit of having a universal labelling of the channels starts to emerge for deterministic rendezvous algorithms like the JS algorithm.

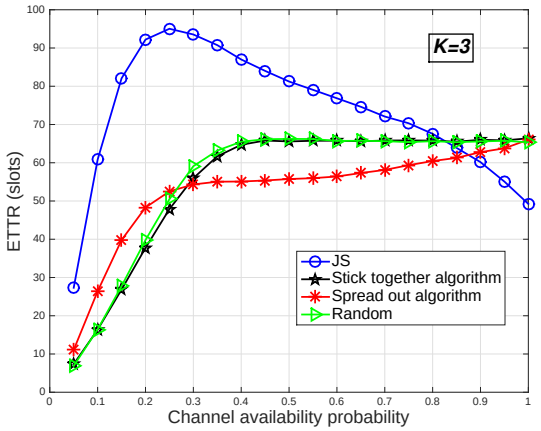

Fig. 5: Comparison of the ETTRs of the four multiuser rendezvous algorithms for a system of three users. The channel availability probability of each user is the same.

In Figure 6, we also show the simulation results for six users with the same channel availability probability v. The comparison results for six users in Figure 6 are in line with those for three users in Figure 5.

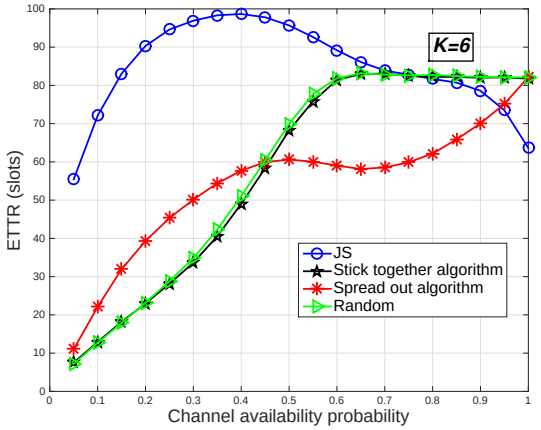

Fig. 6: Comparison of the ETTRs of the four multiuser rendezvous algorithms for a system of six users. The channel availability probability of each user is the same.

We also note that when all the channels are available to every user, i.e., $v = 1$, then both the ETTRs of the stick together algorithm and the spread out algorithm in Figure 5 and Figure 6 are almost the same as that of the random algorithm. From the simulation results, it seems that the ETTR of the random algorithm with K users and N channels can be well approximated by $2N(1 - 1/K)$ for $v = 1$ and large N. To see the intuition behind such an approximation, note that for large N the rendezvous probability for a group of more than two users is $O(N^{-2})$ and thus one only needs to consider the case that each rendezvous occurs with exactly two users. As there are $K(K-1)/2$ pairs of two users, the first time that a rendezvous of two users occurs can be well approximated by a geometric distribution with the parameter $K(K-1)/2N$. Then the expected time for such an event to occur is $2N/K(K-1)$. After the occurrence of such an event, the number of users is reduced from K to $K-1$. A simple induction then shows the ETTR of the random algorithm with K users can be well approximated by

$$\frac{2N}{K \cdot (K-1)} + \frac{2N}{(K-1) \cdot (K-2)} + \ldots + \frac{2N}{2 \cdot 1} = 2N(1 - \frac{1}{K}).$$

Such an approximation argument for the ETTR of the random algorithm is closely related to the expected coalescent time in the Wright-Fisher model that finds the most recent common ancestor for a population [41].

In our second experiment for the multiuser rendezvous algorithms, we allow the channel availability probabilities of the K users to be different. Specifically, the channel availability probability of user i is $V_i/20$, where V_i is uniformly chosen from the set of integers $\{1, 2..., 20\}$ in each simulation. Each data point is again obtained by averaging over 100,000 simulations. As we show in Figure 7, the spread out algorithm does have a better ETTR than the other two algorithms for $K = 2, 3, \ldots, 20$ in this experiment. This is because the channel availability probability of each user is now selected independently and the probability that all the users have very small channel availability probabilities is small. On the other hand, we also evaluate the JS algorithm [11] for this simulation setting. As clearly shown in Figure 7, our two algorithms significantly outperform the JS algorithm in terms of ETTR. The intuition behind this is that the available channel sets of the K users are quite different when the channel availability probability is randomly selected by each user. Under such a scenario, having a universal labelling of the channels does not help too much for deterministic algorithms like the JS algorithm. One interesting observation from Figure 7 is that the ETTR is not always increasing in the number of users K. This appears to be quite counterintuitive at the beginning. But, as mentioned before, one factor that affects the TTR is the speed to reach a consensus of the state information. Increasing the number of users increases the speed to synchronize the state information. In particular, when $K \geq 10$, the number of common channels among the K users is reduced to 1 (with a very high probability) and that common channel (channel 0 in our experiment) can be quickly identified by a user after a few pairwise rendezvous. As such, increasing the number of users

does not necessarily increase the ETTR under the assumption that there is at least one common channel.

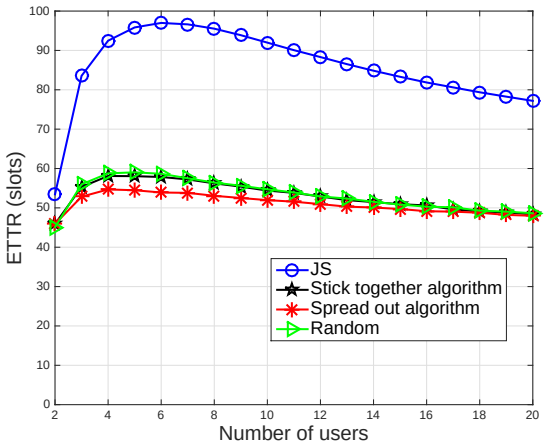

Fig. 7: Comparison of the ETTRs of the four multiuser rendezvous algorithms for a system of K users. The channel availability probability of each user is selected independently.

V. Conclusion

In this paper, we considered the most challenging oblivious rendezvous problem as in [19], [26]. For such a multichannel rendezvous problem, it is generally assumed that (i) there are no distinguishable roles of users, (ii) users' clocks may not be synchronized, (iii) users may have different available channel sets, and (iv) there is no universal labelling of the channels available to the users. By assuming there is at least one common channel in (A1) and there is a unique ID for each user in (A2), we proposed the two-prime modular clock algorithm in Algorithm 4 based on the new class of strong symmetrization mappings. The ETTR of the two-prime modular clock algorithm is almost the same as that of the random algorithm and its MCTTR is still upper bounded by a finite constant comparable to the best bound in the literature. As the ETTR of the random algorithm is close to the lower bound of the oblivious rendezvous problem, the ETTR of our algorithm is nearly optimal. We also extended the two-prime modular clock algorithm for multiuser rendezvous. For this, we proposed the stick together algorithm in Algorithm 5 and the spread out algorithm in Algorithm 6. One interesting finding for the multiuser rendezvous problem is that the spread out algorithm is not always better than the stick together algorithm as commonly claimed in the literature.

We note that there are other methods to achieve multiuser rendezvous. For example, one might consider a two-step approach. First, users collect the state information from other users through a sequence of pairwise rendezvous (or a centralized server). Once all the users share the same state information, they can use such information to achieve multiuser rendezvous. Such a two-step approach is beyond the scope of this paper and will require further study.

We also note that message exchange (in our multiuser rendezvous algorithms) relies heavily on the underlining physical networks. Like most multichannel rendezvous papers in the literature, we did not consider the cost of message exchange in our performance evaluation for our rendezvous algorithms. Understanding the effect of the underlining physical networks to our algorithms (as well as other rendezvous algorithms) will be another important topic for our future research in this area.

References

[1] P. Bahl, R. Chandra, J. Dunagan, "SSCH: slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks," in *Porc. ACM MobiCom*, 2004.

[2] L. DaSilva and I. Guerreiro, "Sequence based rendezvous for dynamic spectrum access," in *Proc. IEEE Int'l Symp. New Frontiers in Dynamic Spectrum Access Networks (DySPAN'08)*, pp. 1-7, Oct. 2008.

[3] J. Mo, H.-S.W. So, and J. Warland, "Comparison of multichannel MAC protocols," *IEEE Transactions on Mobile Computing*, vol. 7, no. 1, pp. 50-65, 2008.

[4] Y. R. Kondareddy and P. Agrawal, "Synchronized MAC protocol for multi-hop cognitive radio networks," in *Proc. IEEE ICC*, 2008.

[5] K. Bian, J.-M. Park, and R. Chane, "A quorum-based framework for establishing control channels in dynamic spectrum access networks," in *Proc. ACM MobiCom*, 2009.

[6] C.-F. Shih, T. Y. Wu, and W. Liao, "DH-MAC: A dynamic channel hopping MAC protocol for cognitive radio networks," in *Proc. IEEE ICC*, 2010.

[7] J. Shin, D. Yang, and C. Kim, "A channel rendezvous scheme for cognitive radio networks," *IEEE Communications Letter*, vol. 14, no. 10, pp. 954–956, 2010.

[8] D. Yang, J. Shin, and C. Kim, "Deterministic rendezvous scheme in multichannel access networks," *Electronics Letters*, vol. 46, no. 20, pp. 1402–1404, 2010.

[9] 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.

[10] Y. Zhang, Q. Li, G. Yu and B. Wang, "ETCH: efficient channel hopping for communication rendezvous in dynamic spectrum access networks," in *Proc. IEEE INFOCOM*, 2011.

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

[12] F. Hou, L. X. Cai, X. Shen, and J. Huang, "Asynchronous multichannel MAC design with difference-set-based hopping sequences," *IEEE Transactions on Vehicular Technology*, vol. 60, pp. 1728–1739, 2011.

[13] R. Gandhi, C. C. Wang, and Y. C. Hu, "Fast rendezvous for multiple clients for cognitive radios using coordinated channel hopping," In *Proc. IEEE SECON*, pp. 434–442, 2012.

[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, 2013.

[15] Z, Lin, H. Liu, X. Chu, and Y.-W. Leung, "Enhanced jump-stay rendezvous algorithm for cognitive radio networks," *IEEE Communications Letters*, vol. 17, no. 9, pp. 1742–1745, 2013.

[16] Z. Gu, Q.-S. Hua, Y. Wang, and F. C. M. Lau, "Nearly optimal asynchronous blind rendezvous algorithm for cognitive radio networks," in *Proc. IEEE SECON*, 2013.

[17] G.-Y. Chang and J.-F. Huang, "A fast rendezvous channel-hopping algorithm for cognitive radio networks," *IEEE Communications Letters*, vol. 17, no. 7, pp. 1475–1478, 2013.

[18] Z. Gu, Q.-S. Hua, and W. Dai, "Local sequence based rendezvous algorithms for cognitive radio networks," In *Proc. IEEE SECON*, pp. 194–202, 2014.

[19] Z. Gu, Q. S. Hua, Y. Wang, and F. C. M. Lau, "Oblivious Rendezvous in Cognitive Radio Networks," in *International Colloquium on Structural Information and Communication Complexity*, pp. 165–179, 2014.

[20] S. Chen, A. Russell, A. Samanta, and R. Sundaram, "Deterministic blind rendezvous in cognitive radio networks." in *Proc. IEEE ICDCS*, pp. 358–367, 2014.

[21] L. Yu, H. Liu, Y.-W. Leung, X. Chu, and Z. Lin, "Channel-hopping based on available channel set for rendezvous of cognitive radios." in *Proc. IEEE ICC*, pp. 1573–1579, 2014.

[22] I. H. Chuang, H.-Y. Wu, and Y.-H. Kuo, "A fast blind rendezvous method by alternate hop-and-wait channel hopping in cognitive radio networks," *IEEE Transactions Mobile Computing*, vol. 13, no. 10, pp. 2171–2184, 2014.

[23] 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.

[24] G.-Y. Chang, W.-H. Teng, H.-Y. Chen, and J.-P. Sheu, "Novel channel-hopping schemes for cognitive radio networks," *IEEE Transactions on Mobile Computing,* vol. 13, pp. 407–421, Feb. 2014.

[25] Z. Gu, Q.-S. Hua, W. Dai, "Fully distributed algorithm for blind rendezvous in cognitive radio networks," In *Proc. ACM MobiHoc*, pp. 155–164, 2014.

[26] 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," In *Proc. ACM MobiHoc*, pp. 165–174, 2014.

[27] M. J. Abdel-Rahman, H. Rahbari, and M. Krunz, "Multicast rendezvous in fast-varying DSA network," *IEEE Transactions on Mobile Computing*, vol. 14, no. 7, pp. 1449–1462, 2015.

[28] C.-S. Chang, W. Liao and C.-M. Lien, "On the multichannel rendezvous problem: fundamental limits, optimal hopping sequences, and bounded time-to-rendezvous," *Mathematics of Operations Research,* vol. 40, no. 1, pp. 1-23, 2015.

[29] 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.

[30] L. Chen, S. Shi, K. Bian, and Y. Ji, "Optimizing average-maximum TTR trade-off for cognitive radio rendezvous," In *Proc. IEEE ICC*, pp. 7707-7712, 2015.

[31] M. J. Abdel-Rahman, H. Rahbari, and M. Krunz, "Multicast rendezvous in fast-varying DSA networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 7, pp. 1449-1462, July 2015.

[32] S. Gilbert, F. Kuhn, C. Newport, and C. Zheng, "Efficient communication in cognitive radio networks," In *Proc. ACM PODC*, pp. 119-128, 2015.

[33] C.-S. Chang, W. Liao, and T.-Y. Wu, "Tight lower bounds for channel hopping schemes in cognitive radio networks," *IEEE/ACM Transactions on Netowrking*, vol. 24, no. 4, pp. 2343–2356, 2016.

[34] C.-C. Chou, C.-S. Chang, D.-S. Lee and J. Cheng, "A necessary and sufficient condition for the construction of 2-to-1 optical FIFO multiplexers by a single crossbar switch and fiber delay lines," *IEEE Transactions on Information Theory*, vol. 52, pp. 4519–4531, 2006.

[35] C.-S. Chang, J. Cheng, T.-K. Huang and D.-S. Lee, "Explicit constructions of memoryless crosstalk avoidance codes via C-transform," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 2030–2033, September 2014.

[36] L. L. Peterson and B. S. Davie, *Computer Networks: A Systems Approach*, 4th edition, San Francisco, CA: Morgan Kaufmann Publishers, 2007.

[37] P. Erdös, "Beweis eines satzes von tschebyschef," *Acta Litt. Univ. Sci., Szeged, Sect. Math.*, vol. 5, pp. 194–198, 1932.

[38] M. El Bachraoui, "Primes in the interval [2n, 3n]," *Int. J. Contemp. Math. Sci.*, vol. 1, no. 13-16, pp. 617–621, 2006.

[39] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *System & Control Letters*, vol. 53, pp. 65–78, 2004.

[40] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, Vol. 52, No. 6, pp. 2508–2530, 2006.

[41] M. Nordborg, "Coalescent theory," *Handbook of statistical genetics*, John Wiley & Sons, 2001.

Cheng-Shang Chang (S'85-M'86-M'89-SM'93-F'04) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1983, and the M.S. and Ph.D. degrees from Columbia University, New York, NY, USA, in 1986 and 1989, respectively, all in electrical engineering.

From 1989 to 1993, he was employed as a Research Staff Member with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. Since 1993, he has been with the Department of Electrical Engineering, National Tsing Hua University, Taiwan, where he is a Tsing Hua Distinguished Chair Professor. He is the author of the book Performance Guarantees in Communication Networks (Springer, 2000) and the coauthor of the book Principles, Architectures and Mathematical Theory of High Performance Packet Switches (Ministry of Education, R.O.C., 2006). His current research interests are concerned with network science, big data analytics, mathematical modeling of the Internet, and high-speed switching.

Dr. Chang served as an Editor for Operations Research from 1992 to 1999, an Editor for the *IEEE/ACM TRANSACTIONS ON NETWORKING* from 2007 to 2009, and an Editor for the *IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING* from 2014 to 2017. He is currently serving as an Editor-at-Large for the *IEEE/ACM TRANSACTIONS ON NETWORKING*. He is a member of IFIP Working Group 7.3. He received an IBM Outstanding Innovation Award in 1992, an IBM Faculty Partnership Award in 2001, and Outstanding Research Awards from the National Science Council, Taiwan, in 1998, 2000, and 2002, respectively. He also received Outstanding Teaching Awards from both the College of EECS and the university itself in 2003. He was appointed as the first Y. Z. Hsu Scientific Chair Professor in 2002. He received the Merit NSC Research Fellow Award from the National Science Council, R.O.C. in 2011. He also received the Academic Award in 2011 and the National Chair Professorship in 2017 from the Ministry of Education, R.O.C. He is the recipient of the 2017 IEEE INFOCOM Achievement Award.

Cheng-Yu Chen received his B.S. degree in the Department of Electronic Engineering and Computer Science at National Tsing Hua University, Hsinchu, Taiwan, in 2015. He is currently an MS student in the Institute of Communications Engineering at National Tsing Hua University, Hsinchu, Taiwan.

Duan-Shin Lee (S'89-M'90-SM'98) received the B.S. degree from National Tsing Hua University, Taiwan, in 1983, and the MS and Ph.D. degrees from Columbia University, New York, in 1987 and 1990, all in electrical engineering. He worked as a research staff member at the C&C Research Laboratory of NEC USA, Inc. in Princeton, New Jersey from 1990 to 1998. He joined the Department of Computer Science of National Tsing Hua University in Hsinchu, Taiwan, in 1998. Since August 2003, he has been a professor. He received a best paper award from the Y.Z. Hsu Foundation in 2006. His current research interests are social networks, network science, game theory and data science. He is a senior IEEE member.

Wanjiun Liao received the PhD degree in Electrical Engineering from the University of Southern California, Los Angeles, CA, USA, in 1997. She is a Distinguished Professor of the Electrical Engineering Department, National Taiwan University, Taipei, Taiwan, and the Director General of the Engineering and Technologies Department, Ministry of Science and Technology (MOST), Taiwan. Her research interests include the design and analysis of wireless networking, cloud networking, and green communications. Prof. Liao was on the Editorial Boards of IEEE Transactions on Wireless Communications and IEEE Transactions on Multimedia. She is very active in IEEE and IEEE ComSoc, including serving as IEEE ComSoc Distinguished Lecturer (2011-2012), IEEE Fellow Committee (2013-2015), IEEE ComSoc Fellow Evaluation Committee (2016-2018), the IEEE ComSoc Director for Asia Pacific Region (2014-2015), and IEEE ComSoc Board of Governors Member-at-Large (2017-2019). She received many recognitions and honors for her achievements from different organizations. She is a Fellow of the IEEE.