How Local Information Improves Rendezvous in Cognitive Radio Networks

***Abstract*—Cognitive Radio Network (CRN) is a promising technique aiming at solving the wireless spectrum scarcity problem. Rendezvous is the fundamental process of CRN. We are devoted to design faster rendezvous algorithms for CRN. We find that local information such as user's ID and the label of an available channel is very useful for designing faster rendezvous algorithms. First, we propose an Asynchronous Anonymous Non-oblivious Global-channel-based (AANG) algorithm. AANG algorithm can guarantee rendezvous for any two users i, j in $2P^2 + 2P$ timeslots, where P is the least prime not less than the total number of channels in the network. Second, we utilize the user's identifier (ID) to design an Asynchronous Non-anonymous Oblivious Local-channel-based (ANOL) algorithm. ANOL algorithm can guarantee rendezvous for any two users i and j in $(l+1)(P_i+2)(P_j+2)$ timeslots, where P_i and P_j are the smallest primes which are not less than the numbers of available channels of user i and j respectively. Third, we propose an Asynchronous Anonymous Non-obvious Local-sequence-based (AANL) rendezvous algorithm which can guarantee rendezvous for any two users in $((P_i+2)(P_j+2)+P)(\lceil log_2 N\rceil+1)$ timeslots, where N is the overall amount of channels in the network and P is the least prime which is not less than N. All of our algorithms can be used in multi-user scenario. We conduct plenty of experiments comparing our algorithms with state-of-the-art rendezvous algorithms under different scenarios, which corroborate our analysis.**

***Index Terms*—Cognitive Radio Network, Local Information, Rendezvous Algorithms, Channel Label, identifier**

I. Introduction

The wireless spectrum is invaluable for the transmission of wireless signals because the resource of wireless spectrum is finite. The wireless spectrum is divided into licensed spectrums and unlicensed spectrums. However, the overcrowding problem of the unlicensed spectrums is becoming more and more severe due to the rapid growth of wireless devices. On the other hand, the utilization rate of the licensed spectrums is relatively low. Cognitive Radio Network (CRN) is a promising solution for this problem. In CRN, there exist two kinds of users, primary users (PUs) and secondary users (SUs). A PU owns one or more licensed channels while a SU needs to search for "empty" licensed spectrums which are not occupied by PUs [1] to complete communication task.

CRN contains many important stages such as broadcasting, routing, data gathering, neighbor discovery and so on. *Rendezvous* is the fundamental task of CRN, which targets finding a common frequency band (channel) for different users to communicate on. Before rendezvous completes, a user may even not aware of the existence of another user, let alone exchange information.

[1]Unless specifically pointed out, "users" in the rest of this paper refer to SUs.

To achieve rendezvous between users, many solutions have been proposed. Some works requires a central controller or a common control channel (CCC) to achieve rendezvous. However, it is not practical to establish a central controller to control all the users in large-scale networks. And if the number of users exceeds the capacity of the central controller or CCC, this method will become invalid. And it is not safe from the perspective of information security, for the probability of adversary attacks. Hence, some researchers proposed some algorithms without central controller or CCC, which are called blind rendezvous algorithms. Blind rendezvous algorithms are more practical than CCC based rendezvous algorithms. Channel Hopping (CH) technique is widely used in blind rendezvous algorithms. In CH algorithms, users hop to channels according to predefined sequences.

Existing blind rendezvous algorithms can be categorized into different categories as follows.

(i) Synchronous/asynchronous algorithms. Synchronous algorithms suppose there exists a global clock and every user in the network starts rendezvous process at the same time. Asynchronous algorithms don't require a global clock and every user can start rendezvous process at any time.

(ii) Anonoymous/non-anonymous algorithms. Anonymous algorithms don't require every user to have a unique ID. Non-anonymous algorithms suppose there exists a unique identifer (ID) for every user and take use of a user's ID to generate the channel hopping sequence.

(iii) Oblivious/non-oblivious algorithms. Oblivious algorithms don't require there exists a global labeling of all the channels and every user labels channels in the same way. Non-oblivious algorithms require global labeling of channels and leverages it to guarantee rendezvous.

(iv) Global-sequence-based/local-sequence-based algorithms. Global-sequence-based algorithms construct channel hopping sequences containing all the channels. If a channel is not in the available channel set of a user, replace it with one available channel. Local-sequence-based algorithms generates channel hopping sequences utilizing only the local available channels as if a user is not aware of the existence of other channels.

(v) Single-radio/multi-radio algorithms. Single-radio algorithms suppose every user in the network is equipped with only one cognitive radio. Hence, one user can once can only hop to one channel. Multi-radio algorithms assume every user in the network is equipped with more than one cognitive

TABLE I
Comparisons between rendezvous algorithms

Algorithm	MTTR	RD	Time	Anonymous or Non-Anonymous	Oblivious or Non-Oblivious	Sequence	Single Radio or Multi-radio
JS	3NP(P-G) +3P	100%	Asynchronous	Anonymous	Non-Oblivious	Global	Single-radio
DRDS	$3P^2 + 2P$	100%	Asynchronous	Anonymous	Non-Oblivious	Global	Single-radio
CBH	$2l_pP^2$	100%	Asynchronous	Non-Anonymou	Oblivious	Local	Single-radio
HH	$3\|C_a\|\|C_b\|$	100%	Asynchronous	Anonymous	Non-Oblivious	Local	Single-radio
Adv.rdv-l	$(g + log_2g)P_iP_j$	100%	Asynchronous	Non-Anonymous	Oblivious	Local	Single-radio
Adv.rdv-η_1	$(2g + 3)P_iP_j$	100%	Asynchronous	Non-Anonymous	Oblivious	Local	Single-radio
Adv.rdv-η_2	$(g + \lceil\sqrt{g}\rceil\ (2$ $+ \lceil log_2g\rceil)P_iP_j$	100%	Asynchronous	Non-Anonymous	Oblivious	Local	Single-radio
MTP	$2(max\{m_a, m_b\})^2$ $\times 32(\lceil loglog\ n\rceil + 1)$	100%	Asynchronous	Anonymous	Oblivious	Local	Multi-radio
AANG(this paper)	$2P^2 + 2P$	$< 100\%$	Asynchronous	Anonymous	Oblivious	Global	Single-radio
ANOL(this paper)	$(l + 1)(P_i + 2)(P_j + 2)$	100%	Asynchronous	Non-Anonymous	Oblivious	Local	Single-radio
ANOL(this paper)	$((P_i + 2)(P_j + 2) + P)$ $\times(\lceil log_2N\rceil + 1)$	$< 100\%$	Asynchronous	Anonymous	Oblivious	Local	Single-radio

radios. Hence, one user can hop to one or more channels simultaneously.

Rendezvous algorithms can be judged according to some common metrics, such as *Expected Time To Rendezvous (ETTR)*, *Maximum Time To Rendezvous (MTTR)*, *Average Time To Rendezvous (ATTR)*, *Rendezvous Degree (RD)*. *Time To Rendezvous (TTR)* is the number of timeslots consumed from the last user starts rendezvous until the completion of rendezvous. ATTR is the average TTR of an algorithm. MTTR is the maximum TTR needed to achieve rendezvous. In other words, MTTR is the TTR in the worst case. It is obvious that two users can have more than one common channel, RD is the ratio of the least number of different rendezvous channels to the total number of common channels between any two users. It is obvious that the larger RD is, the more channels two users can rendezvous on. Because a channel can be not accessible to a user for some reasons, a larger ID can increase the probability of rendezvous between users. In other words, the larger RD is, the better.

In this paper, we introduce some simple but time-efficient rendezvous algorithms under different scenarios. The following are the main contributions of our paper:

1) We propose a non-anonymous local-sequence-based rendezvous algorithm which outperforms state-of-the-art non-anonymous local-sequence-based rendezvous algorithms.
2) We propose a two-radio local-sequence-based rendezvous algorithm which outperforms the state-of-the-art two-radio local-sequence-based rendezvous algorithm.
3) We formulate the concept of semi-synchronous model and propose a semi-synchronous global-sequence-based rendezvous algorithm which is feasible and has a MTTR of N, which is least in all the existing rendezvous algorithms.
4) We conduct lots of experiments to compare our algorithms with state-of-the-art algorithms. The results show that our algorithms outperforms state-of-the-art rendezvous algorithms.

The rest of this paper is organized as follows. Section II introduces the background and some related works. Section III introduces different categories of models for rendezvous problem and problem formulation. Section IV presents a non-anonymous local-sequence-based rendezvous algorithm and analyzes its performance. Section V presents a two-radio local-sequence-based rendezvous algorithm and analyzes its performance. Section VI presents a semi-synchronous global-sequence-based rendezvous algorithm and analyzes its performance. Simulation results are displayed in Section Section VII. And we conclude this paper in Section VIII.

II. Background and Related Works

A. Prime number and Co-prime Numbers

Prime number is a very important concept in number theory. We first introduce the definition of prime number:

Definition 2.1: A prime number is a natural number which is larger than 1 and can be divided with no remainder only by 1 and itself.

The smallest prime number is 2. There is an important theory about prime number:

Theorem 1: If n is a positive integer and $n \geq 2$, there must exists at least one prime number between n and 2n.

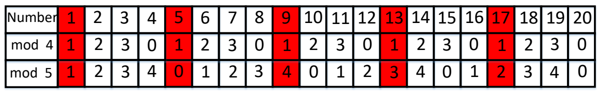
Then we introduce the definition of composite number:

Definition 2.2: A composite number is a natural number which is larger than 1 and has factors besides 1 and itself.

The smallest composite number is 4. There is a property about prime and composite numbers:

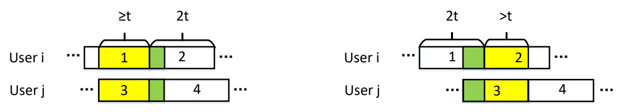
Lemma 2.1: Any positive integer larger than 1 is either a prime number or a composite number.

Then we introduce the definition of co-prime numbers:



Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
mod 4	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0
mod 5	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0

Fig. 1. An example of Theorem 2.



(a) case 1 (b) case 2

Fig. 2. An example of different cases of timeslot overlapping.

Definition 2.3: Two nonzero natural integers a and b are said to be co-prime if the only common factor which could divide them with no remainder is 1.

Co-prime numbers have many important properties:

Lemma 2.2: If a and b are co-prime, the least common multiple of them is their product: $a \times b$.

Lemma 2.3: If a and b are two consecutive positive integers, a and b are co-prime.

Lemma 2.4: If a and b are two consecutive positive odd numbers, a and b are co-prime.

Lemma 2.5: Suppose a is a prime number and b is a composite number. If a is larger than b, they are co-prime. If b is larger than a, but b is not a multiple of a, they are co-prime.

We then introduce an important theorem for rendezvous:

Theorem 2: If m and n are co-prime numbers, then for any integer a, the integers $a, a+n, a+2n, \cdots, a+(m-1)n$ are m distinct numbers under modulo-m arithmetic.

Proof: Choose any two integers from $a, a+n, a+2n, \cdots, a+(m-1)n$, the absolute value of their difference is kn, where $0 < k < m$. If kn mod m equal 0, then $kn = lm$, where l is a positive integer. Then kn or lm is a common multiple of m and n. From Lemma 2.2 we know mn is the least common multiple of m and n, and because $0 < k < m$, kn is a common multiple of m and n that is less than mn, which is a contradiction. Hence, Theorem 2 holds. ■

Fig. 1 illustrates an example of Theorem 2. In Fig. 1, the first row is a string of numbers starting from 1. The second row is the results of these numbers under modulo 4 arithmetic. The third row is the results of these numbers under modulo 5 arithmetic. We can clearly see that 1, 5, 9, 13, 17 all correspond to 1 under modulo 4 arithmetic, but correspond to 1, 0, 4, 3, 2 respectively under modulo 5 arithmetic, which are all the possible results under modulo 5 arithmetic. Theorem 2 plays an important role in rendezvous algorithms and is the main tool we use to design the following algorithms.

B. Related Works

III. Models and Problem Formulations

In this section, we will introduce the foundation model and some other models which are based on the foundation model. Models are of great importance in the design of rendezvous algorithms, because models constrain the resources that we can use. Hence, we need to design different algorithms under different models to achieve high performance for rendezvous process. We will also give the formulation for rendezvous problem in this section.

A. Foundation Model

Suppose there are M users in the network. Suppose the licensed spectrum is divided into n channels which are non-overlapping. Denote the set of channels as $U = \{1, 2, \cdots, n\}$, where n is the total number of channels in U. We assume that every user in the network is equipped with a CRN. User i can sense a set of channels as $C_i = \{c_{i1}, c_{i2}, \cdots, c_{in_i}\}$, where n_i is the number of channels in set C_i. A channel is available to user i when it is not occupied by a PU and it is in C_i. User i has a set of available channels as $V_i = \{v_{i1}, v_{i2}, \cdots, v_{im_i}\}$, where m_i is the number of channels in set V_i. For simplicity, we suppose that V_i doesn't change during the rendezvous process. Let the length of every timeslot be 2t, where $t = 10ms$ according to IEEE 802.22.

Because the lengths of all users' timeslots are equal, user j's one timeslot can overlap with at most two consecutive timeslots of user i. Fig. 2 illustrates this problem in two cases. Suppose timeslot 3 is one timeslot of user j. And the left timeslot of user i which overlaps with timeslot 3 is timeslot 1. The right timeslot of user i which overlaps with timeslot 3 is timeslot 4. The yellow part stands for the overlapping part of timeslot 1 and 3, while the green part stands for the overlapping part of timeslot 2 and 3. In Fig. 2(a), the length of the yellow part is greater than or equal to t. In Fig. 2(b), the length of the green part is greater than t. There is a special case that the length of the yellow part or the green part is zero, which means that the timeslots of user i and user j are aligned. Hence, no matter when the users start their rendezvous process, the maximum overlap of their timeslots is at least t, which is enough for two users to find each other and exchange information.Foundation model is the basis of other models we will introduce in the following subsections.

B. Problem Formulations

In this paper, we focus on designing rendezvous algorithms for two-user scenario, because it is the fundamental process for multiple-user rendezvous. Two-user rendezvous algorithms can be expanded into multiple-user rendezvous algorithms. Then we give the formulations of the three problems we will solve:

Problem 1: Utilize an arbitrary available channel of a user to design an asynchronous anonymous non-oblivious global-sequence-based rendezvous algorithm which has low MTTR and high RD.

Problem 2: Utilize the ID of a user to design an asynchronous non-anonymous oblivious local-sequence-based rendezvous algorithm which has low MTTR and highest RD.

Problem 3: Utilize an arbitrary available channel of a user to design an asynchronous anonymous non-oblivious local-sequence-based rendezvous algorithm which has low MTTR and high RD.

IV. A Global-sequence-based Rendezvous Algorithm Based on the Channel Label

In this section, we will propose a global-sequence-based rendezvous algorithm which is based on the channel label.

We first propose an algorithm to generate the subsequences to be used.

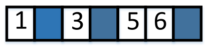
Algorithm 1 Subsequence Generating Algorithm

```
 1: Input: the set of available channels $V = \{v_1, v_2, \cdots, v_m\}$,
    and length $L$;
 2: Initialize array $S$ with length $L$;
 3: Initialize $i := 0$;
 4: while $i < L$ do
 5:     if $i \leq m$ then
 6:         $S[i] := v_i$;
 7:     else
 8:         Randomly choose a channel from $V$, denote it as $c$;
 9:         $S[i] := c$;
10:     end if
11:     $i := i + 1$;
12: end while
13: Output: Sequence $S$.
```

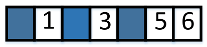
The main idea of Alg.1 is to generate a subsequence S with length L according to V. The process is rather simple. We fill the first m elements in S with the channels in V. For the rest $(L - m)$ elements, we randomly pick channels from V to fill them.

Algorithm 2 Global-Sequence-Based Non-Oblivious Rendezvous Algorithm

```
 1: Input: the set of available channels $V = \{v_1, v_2, \cdots, v_m\}$
    and the total number $N$ of all channels;
 2: Find the least prime $P$ that ensure $P \geq N$;
 3: Randomly choose a channel from $V$, denote it as channel
    $(c)$;
 4: Initialize $t := 0$;
 5: while $0 \leq t < 2P$ do
 6:     Access channel $(c)$;
 7:     $t := t + 1$;
 8: end while
 9: Invoke Alg.1 to generate a subsequence $S$ with channel
    set $V$ and length $P$;
10: while Not rendezvous do
11:     if $t \neq 2P$ and $t \bmod (2P) = 0$ then
12:         Rotate $S$ to the right by $c$ steps;
13:     end if
14:     Access channel $S[t \bmod P]$;
15:     $t := t + 1$
16: end while
```



(a) Original subsequence



(b) The first rotated subsequence

Fig. 3. An example of generated subsequence when $V = \{1, 3, 5, 6\}$, $N = 6$ and the first rotated subsequence.

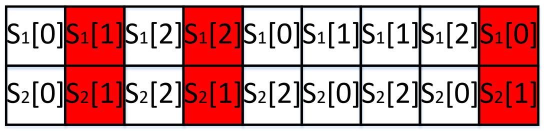

S1[0] S1[1] S1[2] S1[2] S1[0] S1[1] S1[1] S1[2] S1[0]
S2[0] S2[1] S2[2] S2[1] S2[2] S2[0] S2[2] S2[0] S2[1]

Fig. 4. An example of Theorem 3.

In Alg. 2, we first find the least prime which is not less than the total number of all channels. Then we randomly choose a channel from V, denote it as channel (c), and access this channel for the first $2P$ timeslots. We call this the **first stage** , the following is the **second stage** .

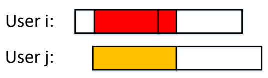
Then, we will generate a subsequence S with length P. If channel (a) is in the available channel set V, the $a - th$ channel in S is channel (a). For the channels that are not exist in V, their places will be replaced by channels which are randomly picked from V. And the last $(P - n)$ channels in S are randomly picked from V. The period is $2P$ timeslots. Then we will rotate the sequence by c steps to the right every period. In every period, we will hop to channels according to rotated sequence.

Fig.3 illustrates an example of the original generated subsequence and the first rotated subsequence. In this example, $V = \{1, 3, 5, 6\}$ and $n = 6$, so $P = 7$. We generate a sequence as Fig.3(a), where the blue rectangles correspond to randomly picked channels. And a rectangle with number in it, denote it as i, represents the Channel (i). Because the least label in V is 1, so the sequence is rotated by 1 step to the right every period. The first rotated sequence is shown in Fig.3(b).

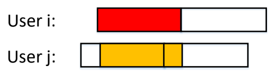
We then introduce a theorem which will be used in the later discuss.

Theorem 3: Suppose there are two aligned subsequences S_1 and S_2 with identical length P and P is a prime. Suppose $0 \leq k_1 \leq P, 0 \leq k_2 \leq P$ and $k_1 \neq k_2$. If in every period we rotate S_1 and S_2 to the right by k_1 and k_2 steps respectively and concatenate them to S_1 and S_2 respectively. Then after P periods, every element of S_1 will meet every element of S_2 and vice versa.

Proof: Because S_1 and S_2 are aligned, then $S_1[i]$ will meet $S_2[i]$ in the first period, where $0 \leq i < P$. Without lose of generality, suppose $k_1 > k_2$. Let $k_1 - k_2 = m$, then in the later $(P - 1)$ periods, $S_1[i]$ will meet $S_2[(i + m) \bmod P]$, $S_2[(i + 2m) \bmod P]$, $\cdots$, $S_2[(i + (P - 1)m) \bmod P]$ respectively. According to Theorem 2, $i, i+m, i+2m, \cdots, i+(P-1)m$ are P distinct numbers under modulo-P arithmetic. Hence, $S_1[i]$ meets all the elements of S_2 in P periods and Theorem 3 holds. ■



(a) Situation 1



(b) Situation 2

Fig. 5. Two situations of periods overlapping of user i and j

Fig.4 illustrates an example of Theorem 3, where the length of S_1 and S_2 is 3. S_1 is rotated to the right by 1 steps every period, and S_2 is rotated to the right by 2 steps every period. We can clearly see that $S_2[1]$ meets $S_1[1], S_1[2], S_1[0]$ in the three consecutive periods.

Theorem 4: The MTTR of Alg.2 is $2P^2 + 2P$.

Proof: We show discuss this problem in two cases.

Case 1 : Suppose two users choose the same channel, denote it as channel c. Without loss of generality, suppose user i starts rendezvous process earlier than user j. For clarity, we divide this case into two subcases.

Subcase 1.1 : Suppose user j starts rendezvous process when user i is still in the first stage, they will both access channel c at the same time. Hence, rendezvous is achieved.

Subcase 1.2 : Suppose user j starts rendezvous process when user i is already in the second stage. Then in the following $2P$ timeslots, user i will access each available channel including channel c at least once, while user j will stay on channel c in the $2P$ timeslots. Hence, rendezvous can be achieved in the $2P$ timeslots.

Case2 : Suppose user i and j choose different channels, denote them as channel (c_i) and channel (c_j). When user i and j both get into the second stage, they will both rotate their sequences according to it. There are two situations of periods overlapping of user i and j, we will discuss them separately.

Subcase 2.1 : Suppose the periods of user i and j are aligned, then the scenario is the same as the setting in Theoren 2 except for that we duplicate the current subsequences before rotate it and attach them to the corresponding sequences. The difference result in double time delay. According to Theorem 3, rendezvous will be achieved in $2P^2 + 2P$ timeslots.

Subcase 2.2 : Suppose the periods of user i and j are not aligned. There are two situations of periods overlapping of user i and j as illustrated in Fig.5. The yellow rectangle in Fig.5(a) represents the first half of user j's period, while the red rectangle represents the corresponding part in user i's period. The red rectangle includes all the elements of subsequence S_i, because the two halves of user i's period are the same rotated subsequence of S_i. Then this situation is similar to Subcase 2.1, and according to Theorem 3, rendezvous will be achieved in $2P^2+2P$ timeslots. Situation 2 can be analysed in the same way, so we omit it here.

Combine these together, we have shown that Theorem 4 holds. ■

Theorem 5: The RD of Alg.2 is 100% if user i and j choose different channels.

Proof: If user i and j choose different channels, according to the case 2 of the proof of Theorem 4, every available channel of V_i will meet every available channel of V_j in $2P^2 + 2P$ timelsots. ■

V. A Local-Sequence-Based Rendezvous Algorithm Utilizing the ID

In this section, we will propose an ID-based asynchronous non-anonymous oblivious rendezvous algorithm.

The main idea of this algorithm is to create a channel hopping sequence composed of $l + 1$ subsequences, where l is the length of ID.

Algorithm 3 Local-sequence-based Non-Anonymous Rendezvous Algorithm

```
1:  Input: the set of available channels $V = \{v_1, v_2, \cdots, v_m\}$,
    and its binary ID whose length is $l$ bits;
2:  Find the smallest prime $P$ that ensure $P \geq m$;
3:  if $P = 3$ then
4:      $P := 5$;
5:  end if
6:  Initialize $t := 0, i := 0, j := 0$;
7:  Invoke Alg.1 to generate three sequences $S_1, S_2, S_3$ with
    channel set V and length $P, P+1, P+2$ respectively;
8:  Find the channel with the least label in $V$, denote it as
    channel $(c)$;
9:  while Not rendezvous do
10:     $i := \lfloor t/(l+1) \rfloor$;
11:     $j := t \bmod (l+1)$;
12:     if $j < l$ then
13:         if ID$[j] = 0$ then
14:             Access channel $S_2[i \bmod (P+1)]$;
15:         else
16:             Access channel $S_3[i \bmod (P+2)]$;
17:         end if
18:     else
19:         Access channel $S_1[i \bmod P]$;
20:     end if
21:     $t := t + 1$;
22: end while
```

In Alg.3, we creates a channel hopping sequence consisting of $(l+1)$ subsequences. The $(l+1)$ subsequences are generated by Alg.1. There are three kinds of subsequences, which are S_1, S_2 and S_3. The lengths of S_1, S_2 and S_3 are P, P+1, P+2 respectively.

Fig.6 illustrates an example of Alg.3. In this example, the length of ID is 5 and user i's ID is 01101. In round 1, the first l elements of round 1 are either $S_2[0]$ or $S_3[0]$. If the corresponding digit of the i-th element in ID is 0, then the i-th element will be $S_2[0]$. If the corresponding digit of the i-th element in ID is 1, then the i-th element will be $S_3[1]$. The last element of round 1 is $S_1[0]$.

Theorem 6: Alg.3 can guarantee rendezvous for two asynchronous users in $(l+1)(P_i+2)(P_j+2)$ timeslots if $V_i \cap V_j \neq \emptyset$.

Proof: We discuss in two cases:

Case 1 :

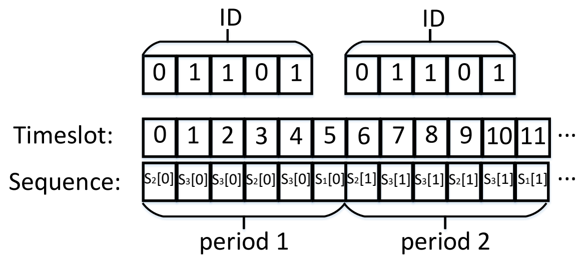
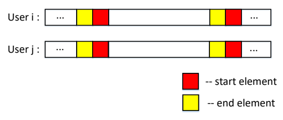
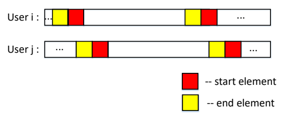


Fig. 6. An example of Alg. 3.



(a) subcase 1



(b) subcase 2

Fig. 7. Two subcases of case 1 and case 2.

Suppose $P_i = P_j$. Let $P = P_i = P_j$ We divide this case into two subcases:

Subcase 1.1 : Suppose user i and j's periods are aligned, as illustrated in Fig.7(a). Because user i and user j are two different users, their ID must be different in at least one digit. Without lose of generality, suppose the k-th digit in ID_i is 0 while the k-th digit in ID_j is 1. Then the k-th element in user i's period corresponds to subsequence S_{2i}, whose length is $P+1$. The k-th element in user j's period corresponds to subsequence S_{3j}, whose length is $P+2$. Then in every period, one element of S_{2i} will meet one element of S_{2j}. According to Lemma 2.3, P+1 and P+2 are co-prime. Then rendezvous can be achieved in $(l+1)(P+1)(P+2)$ timeslots according to Theorem 2, as long as user i and j have at least one common channel.

Subcase 1.2 : Suppose user i and j's periods are not aligned, as illustrated in Fig.7(b). In this case, the last element in user i's period will meet one of the first $(l+1)$ elements of user j's period. The last element in user i's period corresponds to subsequence S_{1i}. The corresponding element of user j corresponds to subsequence S_{2j} or S_{3j}. From Lemma 2.3 and 2.4, we know that P and P+1 are co-prome, P and P +2 are also co-prime. Hence, according to Theorem 2, rendezvous can be achieved in $(l+1)P(P+1)$ or $(l+1)P(P+2)$ timeslots respectively.

Case 2 : Suppose $P_i \neq P_j$. We divide this case into two subcases:

Subcase 2.1 : Suppose user i and j's periods are aligned, as illustrated in Fig.7(a). In this case, the last element in user i's period will meet the last element in user j's period. The last element in user i's period corresponds to subsequence S_{i1}. The last element in user j's period corresponds to subsequence S_{j1}. Because $P_i \neq P_j$, rendezvous can be achieved in $(l+1)P_iP_j$ timeslots according to Theorem 2, as long as user i and j have at least one common channel.

Subcase 2.2 : Suppose user i and j's periods are not aligned, as illustrated in Fig.7(b). Without lose of generality, suppose $P_i > P_j$. In this case, the last element in user i's period will meet one of the first $(l+1)$ elements of user j's period. The last element in user j's period will meet one of the first $(l+1)$ elements of user i's period. The last element in user j's period corresponds to subsequence S_{1j}, whose length is P_j. The corresponding element of user j corresponds to subsequence S_{2i} or S_{3i}, whose lengths are P_i+1 and P_i+2 respectively. Because P_i and P_j are two different primes, $P_i > P_j+2$ or $P_i = P_j+2$. If $P_i > P_j+2$, from Lemma 2.5, we know P_i and P_j+1 are co-prime, P_i and P_j+2 are co-prime. According to Theorem 2, rendezvous can be achieved in $(l+1)P_i(P_j+1)$ or $(l+1)P_i(P_j+2)$ timeslots respectively. If $P_i = P_j+2$, then $P_i+1 = P_2+3$, $P_i+2 = P_j+4$. If $P_j = 2$, $P_i = 2+2 = 4$ is not a prime. In Alg.3, we set P to be 5 if $P = 3$. Because $P_2 \neq 2$ and $P_2 \neq 3$, P_2+3 and P_2+4 can not be divided by P_2 without remainder. From Lemma 2.5, we know that both P_i+1 and P_i+2 are co-prime with P_j. According to Theorem 2, rendezvous can be achieved in $(l+1)(P_i+1)P_j$ or $(l+1)(P_i+2)P_j$ timeslots respectively.

Combine the above together, Theorem 6 holds. ■

Theorem 7: The RD of Alg.3 is 100%.

Proof: For any user, the corresponding subsequence S_1, S_2, S_3 all contains all the channels in its available channel set. From the proof of Theorem 3, we know that one subsequence of user i will meet one subsequence of user j and the lengths of the subsequences are co-prime. Denote the two subsequences as S_i and S_j. From Theorem 2, we know that every element in S_i will meet all of the elements in S_j and vice versa. Hence, the RD of Alg.3 is 100%. ■

VI. A Local-Sequence-Based Rendezvous Algorithm Utilizing the Channel Label

In this section, we will propose a rendezvous algorithm which utilizes the label of a channel in the available channel set.

The main idea of Alg.4 is to randomly choose an available channel c from V and use the binary representation of c as the user's ID. It is obvious that the available channel chosen by two users can be identical, but our algorithm can solve this problem, because the label of an available channel is very useful.

In Alg.4, we divide timeslots into two subsequences, S and C. Subsequence S is further divided into three subsequences S_1, S_2 and S_3. Subsequence C is filled with channel (c). There are three situations of the overlapping of the timeslots of user i and j, as shown in Fig.8

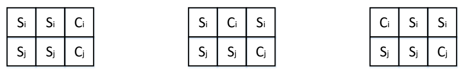
Theorem 8: Alg. 4 can guarantee rendezvous for two asynchronous users in $3(P_i+2)(P_j+2)(\lceil log_2 N\rceil+1)$ timeslots as long as $V_i \cap V_j \neq \emptyset$.

Proof: We discuss this problem in two cases.

Case 1: Suppose the overlapping of timeslots of user i and user j is as shown in Fig.8(a). Then we discuss in two subcases.

Algorithm 4 Channel-Label-Based Local-sequence-based Non-Oblivious Rendezvous Algorithm

```
Input: the set of available channels $V = \{v_1, v_2, \cdots, v_m\}$ and the number of all channels: $N$;
Find the smallest prime $P$ that ensure $P \geq m$;
if $P = 3$ then
    $P := 5$;
end if
Randomly choose a channel $c$ from $V$;
Convert $c$ into a binary string $CH$ whose length is $\lceil log_2 N \rceil$;
Invoke Alg.1 to generate three subsequences $S_1, S_2, S_3$ with channel set $V$ and length $P, P+1, P+2$ respectively.
Initialize $t := 0$, $i := 0$, $j := 0$;
while Not rendezvous do
    if $t < P_N(\lceil log_2 N \rceil + 1)$ then
        Access channel $(c)$;
    else
        $i := \lfloor t/((\lceil log_2 N \rceil + 1)) \rfloor$;
        $j := t$ mod $(\lceil log_2 N \rceil + 1)$;
        if $j < \lceil log_2 N \rceil$ then
            if $CH[j] = 0$ then
                Access channel $S_2[i$ mod $(P+1)]$;
            else
                Access channel $S_3[i$ mod $(P+2)]$;
            end if
        else
            Access channel $S_1[i$ mod $P]$;
        end if
    end if
    $t := t + 1$;
end while
```



(a) Situation 1 (b) Situation 2 (c) Situation 3

Fig. 8. Three situations of timeslot overlapping of user i and j.

Subcase 1.1: If two users choose the same channel, because C_i meets C_j every period and $c_i = c_j$, rendezvous will be achieved in one period, which is 3 timeslots.

Subcase 1.2: If two users choose different channels, then CH_i will be different from CH_j in at least one bit. Then this situation is the same as that in Alg.3, except for the time cost is three times. According to the proof of Theorem 6, rendezvous for user i and user j will be achieved in $3(P_i + 2)(P_j + 2)(\lceil log_2 N \rceil + 1)$ timeslots.

Case 2: Suppose the overlapping of timeslots of user i and user j is as shown in Fig.8(b) or Fig.8(c). We also discuss in two subcases.

Subcase 2.1: If two users choose the same channel, denote it as channel (c). Then channel (c) is in both $S_{ki}(k = 1, 2, 3)$ and $S_{kj}(k = 1, 2, 3)$. Because S_i overlaps with C_j and S_j overlaps with C_i in every period, then C_j will meet all the elements of S_{1i} in $(\lceil log_2 N \rceil + 1)P_j$ periods, which are $3(\lceil log_2 N \rceil + 1)P_j$ timeslots. Hence, rendezvous will be achieved in $3(\lceil log_2 N \rceil + 1)P_j$ timeslots.

Subcase 2.2: If two users choose different channels, then CH_i will be different from CH_j in at least one bit. Because in both Fig.8(b) and Fig.8(c), S_i overlaps with S_j, this situation is the same as that in Alg.3, except for the time cost is three times. According to the proof of Theorem 6, rendezvous for user i and user j will be achieved in $3(P_i + 2)(P_j + 2)(\lceil log_2 N \rceil + 1)$ timeslots.

Combine these together, Theorem 8 holds. ■

Theorem 9: The RD of Alg.4 is 100% if user i and j choose different channels.

Proof: If user i and j choose different channels, then CH_i and CH_j are two different strings. Because S_i overlaps with S_j in all cases, the situation is the same as that in Alg.3. From the proof of Theorem 7, we know that the RD of Alg.4 is 100%. ■

VII. Simulation

VIII. Conclusion

IX. Acknowledgements