EFFICIENT SPECTRUM MANAGEMENT AND RENDEZVOUS SCHEMES FOR COGNITIVE RADIOS

by

Ji Li

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical Engineering

Charlotte

2015

ABSTRACT

JI LI. Efficient Spectrum Management and Rendezvous Schemes for Cognitive Radios. (Under the direction of DR. JIANG (LINDA) XIE)

You must have an abstract. The content of your abstract goes here.

The abstract is the briefest possible summary of the work and conclusions not exceeding one page in length.

CHAPTER 2: Related Work

2.1 Current Rendevous Schemes

CHAPTER 3: Proposed Research

3.1 Communication Framework for Wide-band Cognitive Radio Networks

3.1.1 System Model

We consider that SUs in a CRN can dynamically access a wide-band spectrum which has a total of M channels indexed from 1 to M, where M is a large number which could be hundreds. There are totally N SUs. Each SU is equipped with only one cognitive radio that cannot perform spectrum sensing and data transmission at the same time. Each SU has a unique ID which can be a positive integer. Furthermore, we assume that each SU can obtain its available channel set using a spectrum sensing algorithm [1]. Initially, a SU does not know any information about other SUs, even their unique IDs. A time-slotted system is adopted in this CRN. In each time slot, a SU either hops onto a channel according to its hopping sequence or stays on its current channel to continue the current communication with another SU. Time slots of different SUs are not necessarily synchronized, which is practical and easy to implement. SUs in the CRN attempt to get information about other SUs within its transmission range, send packets to other SUs, or receive packets from other SUs.

In the rest of the paper, we consider the following two scenarios concerning the available channel sets of a SU pair.

Symmetrical model: Any SU pair has exactly the same available channel set. According to [**?**,2,3], the similarity of the available channel sets between two SUs within one hop is over 85%. Thus, this model is reasonable. Moreover, studying this scenario can help us to get solutions for more complicated scenarios.

Asymmetrical model: Any SU pair may have different but overlapping available channel sets. This scenario is more practical because of the dynamics caused by PUs' and SUs' traffic and locations.

3.1.2 The Proposed Framework

The goal of our proposed framework is to design efficient distributed schemes for each SU to guarantee that it can communicate with other SUs successfully in a wide-band CRN. The framework mainly contains three parts: the initialization process, the process to get control information from neighboring SUs, and the process to send packets to a specific SU. Moreover, the framework does not require a common control channel (CCC) for information exchange. The flow charts of these three parts are shown in Fig. 3.1.

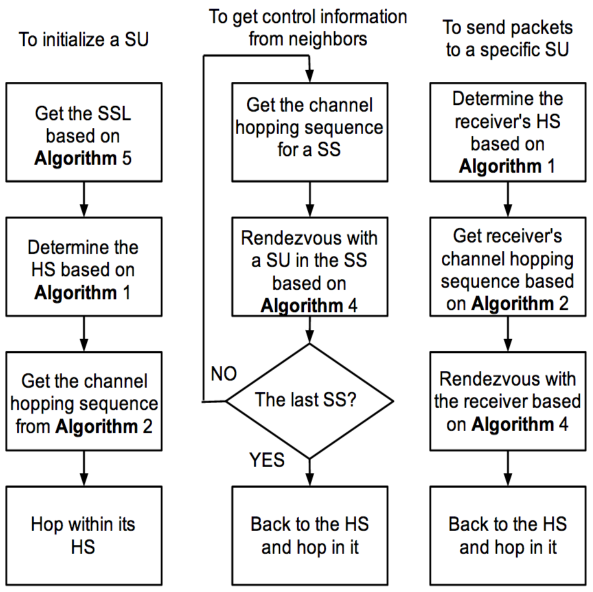

To initialize a SU
Get the SSL based on **Algorithm** 5
Determine the HS based on **Algorithm** 1
Get the channel hopping sequence from **Algorithm** 2
Hop within its HS
To get control information from neighbors
Get the channel hopping sequence for a SS
Rendezvous with a SU in the SS based on **Algorithm** 4
The last SS?
NO
YES
Back to the HS and hop in it
To send packets to a specific SU
Determine the receiver's HS based on **Algorithm** 1
Get receiver's channel hopping sequence based on **Algorithm** 2
Rendezvous with the receiver based on **Algorithm** 4
Back to the HS and hop in it

Figure 3.1: The flow charts of our proposed framework.

During the initialization process, each SU first divides the whole wide-band spectrum into several virtual spectrum segments (SS). Based on **Algorithm 5** (which is explained in Section 3.1.5), each SU can get a unique list containing the number of channels in each SS. We call it the spectrum splitting list (SSL). Using the unique SSL, a SU can obtain the channel index of the start and end channel in each SS (shown in **Algorithm 1**). Then, each SU uses its ID (based on **Algorithm 1**) to determine its home spectrum segment (HS). **Algorithm 1** is based on mapping all the SUs evenly to the whole spectrum, where we assume that the IDs of SUs are evenly distributed within a certain range. A SU hops from a channel to another according to the designed channel hopping sequence (based on **Algorithm 12** which is explained in Section 3.1.3) within its HS. Since SUs are not synchronized, they may be at different positions of the same hopping sequence even they locate in the same HS. This is the initialization process of a SU who just starts in a CRN, as shown in the left of Fig. 3.1.

Algorithm 1 The algorithm to determine a SU's home spectrum's index

```
Input:
    The SU's ID $x$.
Output:
    The SU's home spectrum's index;
1: Use Algorithm 5 to get the SSL $l$;
2: $y = (x - 1)\%M + 1$; //The channel index starts from 1
3: $sum = 0$;
4: for $i = 0$ to $l.length - 1$ do
5:     if $y > sum$ and $y \leq sum + l[i]$ then
6:         return $i$;
7:     end if
8:     $sum += l[i]$;
9: end for
```

After the initialization process, a SU tries to get the control information of all its neighboring SUs which are hopping in the same or different spectrum segments (SSs). We assume that when a SU is not transmitting data packets, it first performs **Algorithm 3** or **Algorithm 4** (which is explained in Section 3.1.3) to rendezvous with a SU in its home spectrum segment (HS) and exchange all their known control information about themselves and others. Through this process, each SU can get all the other SUs' information within the same SS and exchange all of their known information. Therefore, after rendezvous and exchange control information with one existing SU in a SS, a SU can get the control information of all the other SUs in that SS, and all the existing SUs in the SS can also get the information of this SU and all its known information about other SUs. Then, the SU can sequentially hop to each SS, rendezvous with one SU in that SS, and exchange all the known control information about other SUs by executing **Algorithm 3** or **Algorithm 4**. The SU will hop to a next SS only when it has rendezvoused with a SU in the current SS or the time spent in the current SS has already been more than the MTTR of the applied rendezvous scheme (there are no SUs in current SS) . Finally, the SU will obtain and update the control information of all other SUs in the network. A SU can perform the above process periodically to update the control information about other SUs and inform its own change due to the dynamic characteristics of CRNs. The process of getting the control information of all the neighboring SUs is shown in the middle of Fig. 3.1.

After the above process of control information exchange, a SU will get some information about other SUs such as their IDs. When a SU wants to send packets to a specific SU, it first uses the receiver's ID to determine the receiver's home spectrum segment (HS) (based on **Algorithm 1**). Then, the SU sender executes **Algorithm 3** or **Algorithm 4** to realize rendezvous with the SU receiver in the SU receiver's HS for communications. The flow charts of communicating with a specific SU is shown in the right of Fig. 3.1. Because of the spectrum splitting design, the time to obtain control information and to set up communications can be significantly reduced since a SU pair only needs to rendezvous within a specific spectrum segment (SS). In addition, the probability that multiple pairs of SUs rendezvous on the same channel is also reduced since different pairs of SUs may rendezvous in different spectrum segments (SSs). In the following sections, we will give the details of the proposed algorithms in our framework.

3.1.3 Proposed Rendezvous Schemes Under the Symmetrical Model

A SU sender and receiver have exactly the same available channel set under the symmetrical model. This may not be very practical. However, the algorithms developed under this model are the basis for the asymmetrical model.

Under this scenario, each SU has the same available channels in each SS. Each SU hops within its HS according to a channel hopping sequence generated based on its available channel set. Assume that the number of the available channels of both the SU sender and receiver in the SU receiver's HS is n. Since a SU sender and receiver have the same available channel set, we use the indexes of all the available channels in a SS, assuming to be from 1 to n, to design the channel hopping sequence. We desire to get a target sequence $s = s_1 s_2 \dots s_{2n}$ whose length is $2n$ and each channel index from 1 to n appears exactly twice in s. When a SU sender wants to rendezvous with a SU receiver, it first hops to the HS of the SU receiver, generates the same channel hopping sequence as the SU receiver's, and then starts to hop on channel s_1. Both the SU sender and receiver hop according to the sequence sequentially and circularly, which means that when a SU reaches the end of the hopping sequence, it will continue to hop onto the first channel of the sequence in the next time slot. Assume that the SU receiver is initially on channel $s_c, 1 \leq c \leq 2n$. We denote ε as the distance between the SU sender and SU receiver on the channel hopping sequence due to the asynchronization. Therefore, $\varepsilon = c - 1$ or $\varepsilon = 2n - (c - 1)$ when considering the circulation of the channel hopping sequence. We define the rendezvous sequence (RS) as follows:

Rendezvous Sequence (RS): The sequence s is a RS when no matter what ε is, within $2n - 1$ time slots, the SU sender and receiver will hop on a same channel if both of them hop according to the same hopping sequence sequentially and circularly.

We use s to represent our desired rendezvous sequence for the rest of the paper. We induce the following lemma to help us design our rendezvous sequence.

Lemma 1. *Assume that $s_i = s_j = k, i < j$, $1 \leq k \leq n$. If $j - i = k$, s could be a RS.*

Proof. We define the *Sequential Distance* of the element k in the sequence s as $d_k = j - i = k$, and the *Circular Distance* of the element k in the sequence s as $D_k = 2n + i - j = 2n - k$, which is the number of time slots taken to hop from s_j to s_i

circularly in s. We define the distance pair of the channel index k as $p_k = (d_k, D_k)$. Since each d_k is different and bounded in $[1, n]$ and each $D_k = 2n - d_k$ is different and bounded in $[n, 2n-1]$, each p_k is different. Assume that initially the SU sender is on channel s_1 and the SU receiver is on channel s_c Therefore, $\varepsilon = c-1$ or $\varepsilon = 2n-(c-1)$, where $0 \leq \varepsilon \leq 2n-1$. We ignore the case when $\varepsilon = 0$, because under this case, the two SUs are already on the same channel. Thus, we only need to prove our lemma when $1 \leq \varepsilon \leq 2n-1$. Since each p_i is different, we can definitely find a p_t such that $\varepsilon = d_t$ or D_t. Suppose $s_{i1} = s_{j1} = t, i1 < j1$. Thus, when the SU sender hops on channel s_{i1}, it will rendezvous with the SU receiver on channel s_{j1}, or when the SU sender hops on channel s_{j1}, it will rendezvous with the SU receiver on channel s_{i1}. Therefore, sequence s can guarantee a successful rendezvous within $2n-1$ time slots no matter which channel the SU receiver dwells on at the beginning of the rendezvous process. □

In order to efficiently generate s, we notice that s has the following properties.

Lemma 2. *When $n = 4t+2$ or $4t+3$, where t is an integer and $t \geq 1$, the rendezvous sequence does not exist.*

Proof. Designing a rendezvous sequence s as described in **Lemma 1** is equivalent to dividing $1, 2, \ldots, 2n$, these $2n$ numbers into two lists l_l and l_r, each of which has exactly n different numbers, such that for $k = 1, 2, \ldots, n$, $l_r[k] - l_l[k] = k$. Then, we have

$$\sum_{k=1}^{n} l_r[k] - \sum_{k=1}^{n} l_l[k] = \sum_{k=1}^{n} k = \frac{n(n+1)}{2}, \tag{3.1}$$

$$\sum_{k=1}^{n} l_r[k] + \sum_{k=1}^{n} l_l[k] = \sum_{k=1}^{2n} k = n(2n+1). \tag{3.2}$$

Combining (3.7) and (3.8), we can get

$$\sum_{k=1}^{n} l_r[k] = \frac{5n^2+3n}{4}. \tag{3.3}$$

When $n = 4t+2$ or $4t+3$,where t is an integer and $t \geq 1$, the right-hand side of (3.9) is not an integer which contradicts the fact that it is a sum of n integers. □

Lemma 3. *When $n = 4t$ or $4t+1$, where t is an integer and $t \geq 1$, the rendezvous sequence always exists.*

Proof. When $n = 4$, sequence "$1, 1, 4, 2, 3, 2, 4, 3$" satisfies the requirements of our desired rendezvous sequence. When $n = 5$, sequence "$1, 1, 5, 2, 4, 2, 3, 5, 4, 3$" satisfies the requirements of our desired rendezvous sequence. When $n > 5$, the proof can be found in the second page of [4]. □

Based on the proof of **Lemma 2** and **Lemma 17**, we propose **Algorithm 12** to generate our desired rendezvous sequence. Using the channel hopping sequence generated based on **Algorithm 12** and **Lemma 1**, we design the rendezvous algorithm under the symmetrical model as shown in **Algorithm 3**.

We define a round of channel hopping as the length of the $2n$ time slots a SU hops sequentially and circularly according to the channel hopping sequence. An example of rendezvous in one channel hopping round under the symmetrical model is shown in Fig. 3.2. We assume that the number of available channels of a SU pair is 4. We label each available channel using the index from 1 to 4. Based on **Algorithm 12**, their channel hopping sequence is "$1, 1, 4, 2, 3, 2, 4, 3$". Assume that the SU sender starts hopping from the channel whose index is $s_1 = 1$ and the SU receiver currently stays on the channel whose index is $s_3 = 4$. Then, the SU sender and receiver will rendezvous on the channel whose index is $s_4 = 2$ after 3 time slots.

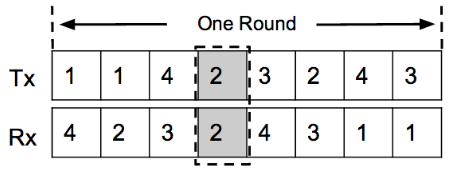


Figure 3.2: An example of rendezvous under the symmetrical model.

Algorithm 2 The algorithm to generate a rendezvous sequence

```
Input:
    The number of channels to generate the channel hopping sequence: $n$
Output:
    The desired rendezvous sequence: $s$
if $n < 4$ or $n\%4 == 2$ or $n\%4 == 3$ then
    $s = \phi$;
    return $s$;
end if
if $n == 4$ then
    $s = [1, 1, 4, 2, 3, 2, 4, 3]$;
    return $s$;
end if
if $n == 5$ then
    $s = [1, 1, 5, 2, 4, 2, 3, 5, 4, 3]$;
    return $s$;
end if
Let $pl = \phi$ be the list of all the position pairs;
$m = \lfloor n/4 \rfloor$;
if $n > 4$ and $n\%4 == 0$ then
    Add all pairs $(4m + r, 8m - r)$, for $r = 0, 1, \ldots, 2m - 1$, to $pl$;
    Add pairs $(2m + 1, 6m)$ and $(2m, 4m - 1)$ to $pl$;
    Add all pairs $(r, 4m - 1 - r)$, for $r = 1, 3, \ldots, m - 1$, to $pl$;
    Add pair $(m, m + 1)$ to $pl$;
    Add all pairs $(m + 2 + r, 3m - 1 - r)$, for $r = 0, 1, \ldots, m - 3$, to $pl$;
end if
if $n > 4$ and $n\%4 == 1$ then
    Add all pairs $(4m + 2 + r, 8m + 2 - r)$, for $r = 0, 1, \ldots, 2m - 1$, to $pl$;
    Add pairs $(2m + 1, 6m + 2)$ and $(2m + 2, 4m + 1)$ to $pl$;
    Add all pairs $(r, 4m + 1 - r)$, for $r = 1, 3, \ldots, m$, to $pl$;
    Add pair $(m + 1, m + 2)$ to $pl$;
    Add all pairs $(m + 2 + r, 3m + 1 - r)$, for $r = 1, 2, \ldots, m - 2$, to $pl$;
end if
for $i = 1$ to $n$ do
    $s[pl[i].first] = s[pl[i].second] = pl[i].second - pl[i].first$;
end for
return $s$;
```

Algorithm 3 The rendezvous algorithm for a SU sender under the symmetrical model

```
Input:
    The available channel set $C$ of the SU sender within the SU sender's HS.
 1: $n$ equals the size of $C$;
 2: if $n\%4 == 2$ or $n\%4 == 3$ then
 3:     $m = 4(\lfloor n/4 \rfloor + 1)$;  /* To guarantee the existence of the RS */
 4: else
 5:     $m = n$;
 6: end if
 7: Generate the channel hopping sequence $s$ based on $m$ and Algorithm 12;
 8: for $k = 1$ to $2m$ do
 9:     if $s_k > n$ then
10:         $s_k = s_k\%n$;  /* To guarantee $s_k$ to be within $n$ */
11:     end if
12: end for
13: $i = 1$;
14: for $j = 1$ to $2m - 1$ do
15:     if The sender receives a CTS packet on channel $C_{s_i}$ of the receiver's HS then
16:         Perform communications;
17:     else
18:         $i++$;
19:     end if
20: end for
```

From **Lemma 1** and **Algorithm 3**, we can induce **Theorem 1**.

Theorem 1. *Under the symmetrical model,* ***Algorithm 3*** *can guarantee rendezvous in one round of the channel hopping. The MTTR of our proposed rendezvous scheme is* $2m-1$ *and the ETTR is* $m-1$*, where* n *is the number of channels in current SS,* $m = n$ *if* $n\%4 = 0$ *or* $n\%4 = 1$*, and* $m = 4(\lfloor n/4 \rfloor + 1)$ *if* $n\%4 = 2$ *or* $n\%4 = 3$*.*

Proof. From **Lemma 1** and **Algorithm 3**, the length of our channel hopping sequence is $2m$ and our rendezvous algorithm can guarantee rendezvous. Thus, according to **Algorithm 3**, the *MTTR* is $2m-1$. When the SU sender is on channel s_1, assume that the SU receiver has equal probability, which is $1/2m$, to be on any channel from s_1 to s_{2m}. According to the proof of **Lemma 1**, the total number of time slots needed to rendezvous of all possible situations is $\sum_{i=0}^{2m-1} i = m(2m-1)$. Thus, the *ETTR* is $\lfloor m(2m-1)/2m \rfloor = m-1$. □

3.1.4 Under the Asymmetrical Model

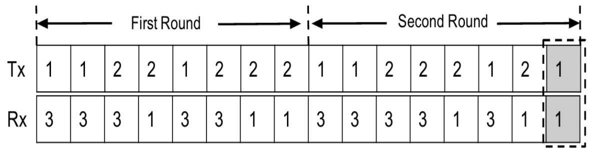
Under the asymmetrical model, the SU sender and receiver have different available channel sets. The challenge here is that before rendezvous happens, the SU sender does not know the available channel set of the SU receiver. Here, we assume that they have at least one common available channel. Otherwise, they can never achieve a successful rendezvous.

As a SU pair may have different available channel sets, each SU's channel hopping sequence should contain all the channels within the SU receiver's HS. We assume that there are totally n channels in the SU receiver's HS which are indexed from 1 to n. The spectrum splitting scheme in Section 3.1.5 can guarantee the existence of the channel hopping sequences. We still use **Algorithm 12** to design the basic channel hopping sequence based on these n channels for both the SU sender and receiver. Additionally, we will replace the unavailable channels in the base channel hopping sequence with the ones randomly chosen from the SU sender's or receiver's own available channels to get their own channel hopping sequences. According to **Lemma 1**, the replacements will not change the property of our channel hopping sequence. Our goal is to design a rendezvous algorithm which can guarantee rendezvous on different channels during several rounds of the channel hopping until rendezvous on the channel which is commonly available for a SU pair. According to the property of the rendezvous sequence, we can simply change the starting position of a SU sender's channel hopping during each channel hopping round. During a new channel hopping round, after changing the distance between the SU pair's positions in the channel hopping sequence, the SU sender and receiver can rendezvous on a new channel. The details of the rendezvous algorithm under the asymmetrical model are shown in **Algorithm 4**.

Algorithm 4 The rendezvous algorithm for a SU sender under the asymmetrical model

```
Input:
    The SU receiver's ID $x$;
    The available channel set $C$ of the SU sender in the SU sender's HS.
 1: Use Algorithm 5 to get the SSL of the whole spectrum band: $l$;
 2: Use Algorithm 1 to get the index of the HS of the SU receiver: $r$;
 3: $n = l[r]$;  /* The index starts from 0 */
 4: Generate the channel hopping sequence $s$ based on $n$ and Algorithm 12;
 5: for $i = 1$ to $2n$ do
 6:   $j = i$;  /* The starting hopping position of the current round */
 7:   for $k = 0$ to $2n - 1$ do
 8:     $\nu = s_j$;
 9:     if $\nu \notin C$ then
10:       Let $\nu$ be a randomly chosen channel in $C$;
11:     end if
12:     if The rendezvous on $\nu$ succeeds then
13:       Perform communications;
14:     else
15:       $j++$;
16:       if $j > 2n$ then
17:         $j = 1$; /* Circularly hopping */
18:       end if
19:     end if
20:   end for
21: end for
```

An example of rendezvous under the asymmetric model is shown in Fig. 3.3. Assume that the available channels of the SU sender are $\{1, 2\}$ and the available channels of the SU receiver are $\{1, 3\}$. Assume that $n = 4$, so the base channel hopping sequence is "$1, 1, 4, 2, 3, 2, 4, 3$". The SU sender starts from channel $s_1 = 1$ and we assume that the SU receiver is on channel $s_3 = 4$. During the first round of channel hopping, after the random replacement process, the channel hopping sequences for the SU sender and SU receiver are "$1, 1, 2, 2, 1, 2, 2, 2$" and "$3, 3, 3, 1, 3, 3, 1, 1$", respectively. According to the channel hopping sequences, the SU sender and receiver do not rendezvous on the same channel for the first round. Then, the SU sender continues the second round of rendezvous by starting from channel $s_2 = 1$ and the base channel hopping sequence will be "$1, 4, 2, 3, 2, 4, 3, 1$". However, the SU receiver will keep using the base rendezvous sequence "$1, 1, 4, 2, 3, 2, 4, 3$" during the second round. Then, during the second round, after the random replacement process, the channel hopping sequences for the SU sender and SU receiver are "$1, 1, 2, 2, 2, 1, 2, 1$" and "$3, 3, 3, 3, 1, 3, 1, 1$", respectively. Finally, the SU sender and receiver rendezvous on channel 1 which is available for both of them at the end of the second round after 15 time slots.



Tx	1	1	2	2	1	2	2	2	1	1	2	2	2	1	2	1
Rx	3	3	3	1	3	3	1	1	3	3	3	3	1	3	1	1

Figure 3.3: An example of rendezvous under the asymmetrical model.

According to the conclusions under the symmetrical model and **Algorithm 4**, we can induce **Theorem 2**.

Theorem 2. *Under the asymmetrical model,* ***Algorithm 4*** *can guarantee rendezvous. The MTTR of the guaranteed rendezvous is* $2n(n - G + 1)$*, and the upper bound of the ETTR is approximately* $(2n - 1)/p$*, where* n *is the number of channels in current SS,*

G is the number of common available channels in current SS, and p is the probability that a channel is available for both the SU sender and receiver.

Proof. The length of the designed sequence s is $2n$. According to **Algorithm 4** and the proof of **Lemma 1**, during each channel hopping round, two SUs can meet on a different available channel. Since there are total G common available channels in current SS, there are at most $n - G + 1$ channel hopping rounds before two SUs hop a common available channel. Therefore, the maximum time slots to a successful rendezvous ($MTTR$) is $2n(n - G + 1)$.

Since the probability that a channel is available for both the SU sender and receiver is p, the probability that a successful rendezvous does not happen until the kth round is $(1 - p)^{k-1}p$. Therefore,

$$\begin{aligned} ETTR &\leq \sum_{k=1}^{2n}(1-p)^{k-1}pk(2n-1) \\ &= \left(\frac{1-(1-p)^{2n}}{p} - 2np^{2n}\right)(2n-1) \\ &\approx \frac{2n-1}{p}. \end{aligned}$$

□

Note that the $ETTR$ of our proposed rendezvous algorithms linearly increases with n. This is much better than other existing rendezvous algorithms under the asymmetrical model whose $ETTR$ is at least $O(n^2)$ such as the ones in [5–7].

3.1.5 Proposed Spectrum Splitting Scheme

First, our proposed spectrum splitting algorithm is based on the following mathematical truth.

Lemma 6. *For any integer $a = 4t + 2, t > 2$, a can be represented as $a = x + y$, where $x = 4u + 1, u \geq 1$ and $y = 4v + 1, v \geq 1$, $a, t, u,$ and v are integers.*

Proof. When $a = 4t + 2, t > 2$, if $t = 2q$, q is an integer and $q \geq 1$, then $a = 8q + 2$, thus $a = (4q + 1) + (4q + 1)$. If $t = 2q + 1, q \geq 1$, then $a = 8q + 6$, thus $a = [4(q + 1) + 1] + (4q + 1)$. □

Lemma 7. *For any integer $a = 4t + 3, t > 2$, a can be represented as $a = x + y + z$, where $x = 4u + 1, u \geq 1$, $y = 4v + 1, v \geq 1$, and $z = 4w + 1$, $a, t, u, v,$ and w are integers.*

Proof. When $a = 4t + 3$, if $t = 3q$, q is an integer and $q \geq 1$, then $a = 12q + 3$, thus $a = (4q + 1) + (4q + 1) + (4q + 1)$. If $t = 3q + 1, q \geq 1$, then $a = 12q + 7$, thus $a = [4(q + 1) + 1] + (4q + 1) + (4q + 1)$. If $t = 3q + 2, q \geq 1$, then $a = 12q + 11$, thus $a = [4(q + 1) + 1] + [4(q + 1) + 1] + (4q + 1)$. □

Lemma 8. *For any integer $a = 4t, t >= 2$, a can be represented as $a = x + y$, where $x = 4u, u \geq 1$ and $y = 4v, u \geq 1$, $a, t, u,$ and v are integers.*

Proof. When $a = 4t, t >= 2$, if $t = 2q$, q is an integer and $q \geq 1$, then $a = 8q$, thus $a = 4q + 4q$. If $t = 2q + 1, q \geq 1$, then $a = 8q + 4$, thus $a = 4q + 4(q + 1)$. □

Lemma 9. *For any integer $a = 4t + 1, t >= 2$, a can be represented as $a = x + y$, where $x = 4u, u \geq 1$ and $y = 4v + 1, v \geq 1$, $a, t, u,$ and v are integers.*

Proof. When $a = 4t + 1, t >= 2$, if $t = 2q$, q is an integer and $q \geq 1$, then $a = 8q + 1$, thus $a = 4q + (4q + 1)$. If $t = 2q + 1, q \geq 1$, then $a = 8q + 5$, thus $a = 4(q + 1) + (4q + 1)$. □

Let θ be the minimum number of channels in a SS. We will discuss how to choose the value of θ in Section **??** through simulation results. Given θ, SUs can execute **Algorithm 5** to determine the SSL.

Based on **Lemma 6** to **Lemma 9** and **Algorithm 5**, each SU can use the same scheme to split the whole spectrum which guarantees the existence of our designed channel hopping sequence in each spectrum segment. An example of our spectrum

Algorithm 5 The algorithm to determine the number of channels in each spectrum segment

Input:
The total number of channels in the whole spectrum band: M; the threshold of the minimum number of channels in each spectrum segment: θ

Output:
The SSL: l

```
1: Initialize $l$ as an empty list;
2: Initialize $Q$ as an empty queue;
3: $Q.push(M)$;
4: while $Q$ is not empty do
5:    Let $\omega$ be the element in $Q$'s head;
6:    Pop $\omega$ out of $Q$;
7:    Use Lemma 6 to Lemma 9 to represent $\omega$ using a set of numbers so that $\omega$
      is the sum of these numbers; Let the set be $A$;
8:    if for each element $\beta \in A$, $\beta \geq \theta$ then
9:       Push all the elements in $A$ into $Q$;
10:   else
11:      Add $\omega$ to $l$;
12:   end if
13: end while
14: return $l$;
```

splitting scheme is that when $M = 100$ and $\theta = 20$, the SSL is $[28, 24, 24, 24]$ and each element in this SSL can guarantee the existence of our rendezvous sequence according to **Lemma 17**.

3.2 Rendezvous Schemes with Multiple Cognitive Radios

3.2.1 System Model And Problem Description

In this paper, we assume that there are totally N SUs and N_p PUs coexisting. There are totally M channels that all the SUs and PUs can access. Each SU is equipped with a GPS which can get its location coordinates. Furthermore, each SU is equipped with exactly R homogeneous half-duplex cognitive radios. Therefore, a cognitive radio cannot send and receive signals simultaneously. We assume that for a SU, all its cognitive radios use the same transmission power and have the same interference range.

There is no CCC in the network through which all SUs can exchange their control information. Before the initialization process of a CRN, each SU does not know any information about other SUs, which is a very practical assumption. A time-slotted system is deployed among all SUs and they do no need to be strictly synchronized.

A channel hopping scheme is adopted for the rendezvous process. During a rendezvous process, each cognitive radio of a SU hops to a channel chosen from its available channel set, senses the current availability of the channel, and sends an RTS packet or waits for an RTS packet from another SU. Therefore, each time slot includes the sensing phase and the sending or receiving phase during the rendezvous process. According to existing rendezvous schemes, a successful rendezvous occurs when two SUs hop to a common available channel at the same time slot, but the process of sending an RTS packet and receiving a CTS reply is not considered. In this paper we define this kind of rendezvous as *channel rendezvous*. This may not be a problem when one SU is always the sender while the other is the receiver, which means that there is an explicit send-or-receive relationship between the two SUs. Under this scenario, after one SU sends an RTS packet, if the other SU receives the RTS packet and replies a CTS packet, the rendezvous succeeds.

However, during the initialization phase of a CRN, each SU may want to rendezvous with other SUs to exchange control information. There is no explicit role for a SU, since it cannot know if other SUs are senders or receivers currently. If two SUs hop to a common available channel and they both have their transceivers on, none of them can receive the RTS packets, which can fail a successful rendezvous. Furthermore, each SU does not know if its target SU is on the current common available channel and is sending or receiving in the current time slot. Practically, only when the two SUs hop to a common available channel and when one SU is sending while the other is trying to receive on that channel, a successful rendezvous can finally happen. In this paper, we denote this kind of rendezvous as *link rendezvous*. Therefore, when a SU is equipped with multiple cognitive radios, during a rendezvous process, the SU should not only consider which available channel each cognitive radio should tune to but also determine that each radio should be in the sending or receiving mode. We call this problem as the *send-or-receive problem* in a rendezvous scheme.

For the ith SU with R cognitive radios, during time slot t of a rendezvous process, it generates a *task allocation list* $\mathbb{A}_i^t = (c_{i1}^t, s_{i1}^t), (c_{i2}^t, s_{i2}^t), \ldots, (c_{iR}^t, s_{iR}^t)$, where $1 \leq c_{iu}^t \leq M$, $s_{iu}^t = 1$ (send) or $s_{iu}^t = 0$ (receive), and (c_{iu}^t, s_{iu}^t) is called a *task pair* for the uth cognitive radio of the ith SU during time slot t. We also denote C_i^t as the available channel set of the ith SU during time slot t. We define *link rendezvous* as follows.

Link Rendezvous: A successful link rendezvous between the ith SU and the jth SU during time slot t requires that there exists a task pair (c_{iu}^t, s_{iu}^t) for the uth cognitive radio of the ith SU and a task pair (c_{jv}^t, s_{jv}^t) for the vth cognitive radio of the jth SU such that $c_{iu}^t \in C_i^t$, $c_{jv}^t \in C_j^t, c_{iu}^t = c_{jv}^t$, and s_{iu}^t XOR $s_{jv}^t = 1$.

Moreover, we can notice that $\mathbb{A}_i^t$ should satisfy the following constraints:

1. No two task pairs (c_{iu}^t, s_{iu}^t) and (c_{iv}^t, s_{iv}^t) satisfy that $c_{iu}^t = c_{iv}^t$ and $s_{iu}^t = s_{iv}^t = 1$, which means that two cognitive radios of the same SU cannot send RTS packets simultaneously on the same channel.

2. No two task pairs (c_{iu}^t, s_{iu}^t) and (c_{iv}^t, s_{iv}^t) satisfy that $c_{iu}^t = c_{iv}^t$ and s_{iu}^t XOR $s_{iv}^t =$ 1, which means that when one cognitive radio of a SU is sending an RTS packet, there should be no other cognitive radio of the same SU that is trying to receive RTS packets on the same channel.

Therefore, the problem is how to generate a *task allocation list* $\mathbb{A}$ for each SU to realize a fast and guaranteed rendezvous considering the *send-or-receive problem.*

3.2.2 Rendezvous Schemes Without Any Information

In this section, we discuss the blind rendezvous problem between two SUs when considering multiple cognitive radios and the *send-or-receive problem.* At the beginning of the initialization phase of a CRN, each SU does not know any information about other SUs. Therefore, each SU can only perform a blind rendezvous process based on its local information to set up links with other SUs. However, previous research works only focus on designing the channel-hopping sequence so that the two SUs can hop to a common available channel at the same time, while ignore the *send-or-receive problem*, especially when each SU is equipped with multiple cognitive radios. Our goal is to generate a *task allocation list* for each SU during each time slot based on an existing channel hopping sequence so that two SUs can achieve a successful *link rendezvous* within bounded time slots.

3.2.3 A Basic Solution

Our proposed basic link rendezvous scheme is to generate a *task allocation list* $\mathbb{A}$ for each SU during each time slot based on an existing channel hopping sequence which can guarantee a successful *channel rendezvous* within bounded time slots. Given a specific channel hopping sequence (e.g., the jump-stay channel hopping sequence in [5]), we denote the channel-hopping sequence for the jth radio of the ith SU as $\mathbb{S}_{ij}$, and the element in $\mathbb{S}_{ij}$ at time slot t as $\mathbb{S}_{ij}^t$. After $\mathbb{S}_{ij}$ is generated, for the task allocation list $\mathbb{A}_i^t$, all the values of $c_{i1}^t, c_{i2}^t, \ldots, c_{iR}^t$ are determined as $c_{i1}^t = \mathbb{S}_{i1}^t, c_{i2}^t = \mathbb{S}_{i2}^t, \ldots, c_{iR}^t = \mathbb{S}_{iR}^t$. Therefore, the next step is to set the binary values of 0 or 1 to $s_{i1}^t, s_{i2}^t, \ldots, s_{iR}^t$ to guarantee that $\mathbb{A}_i^t$ satisfy the constraints explained in Section 3.5.7.

For any $c_{ij}^t, 1 \leq j \leq R$, if c_{ij}^t appears more than once among all the R radios, which means that more than one radio will hop on channel c_{ij}^t during time slot t, the radios allocated to channel c_{ij}^t should wait for RTS packets according to the constraints in Section 3.5.7 Other radios will randomly choose sending or receiving roles to themselves.

Here is an example of the $0-1$ allocation solution. Assume that $R = 5$ and $c_{i1}^t = 1, c_{i2}^t = 3, c_{i3}^t = 3, c_{i4}^t = 2, c_{i5}^t = 1$. Since channel 3 appears twice, $s_{i2}^t = 0$ and $s_{i3}^t = 0$, which means that the second and third radio should wait for RTS packets during the current time slot on channel 3. Since channel 1 also appears twice, $s_{i1}^t = 0$ and $s_{i5}^t = 0$, which means that the first and fifth radio should also wait for RTS packets during the current time slot on channel 1. For the fourth radio, we can randomly set $s_{i4}^t = 1$. Finally, the generated *task allocation list* for the ith SU during time slot t is $\mathbb{A}_i^t = \{(c_{i1}^t = 1, s_{i1}^t = 0), (c_{i2}^t = 3, s_{i2}^t = 0), (c_{i3}^t = 3, s_{i3}^t = 0), (c_{i4}^t = 2, s_{i4}^t = 1), (c_{i5}^t = 1, s_{i5}^t = 0)\}$.

The details of our proposed rendezvous scheme for the ith SU is shown in **Algorithm 6**.

Algorithm 6 The basic rendezvous algorithm considering the send-or-receive problem

```
Input:
    The number of cognitive radios: $R$
    The current time slot: $t$
    A pre-designed channel-hopping sequence $\mathbb{S}$
1: Assume that the $j$th radio starts the channel hopping at time $t_j$;
2: for $j = 1$ to $R$ do
3:     $c_{ij}^t = \mathbb{S}_{t+t_j}$;
4: end for
5: for $j = 1$ to $R$ do
6:     if $c_{ij}^t$ appears more than once then
7:         $s_{ij}^t = 0$;
8:     else
9:         $s_{ij}^t$ is randomly set to be either 0 or 1;
10:    end if
11:    Tune the $j$th radio to channel $c_{ij}^t$;
12:    if $c_{ij}^t \in C_i^t$ and $s_{ij}^t = 1$ then
13:        Send an RTS packet on channel $c_{ij}^t$;
14:    else
15:        Wait for an RTS packet on channel $c_{ij}^t$;
16:    end if
17: end for
```

Based on **Algorithm 6**, we can get **Theorem 10**.

Theorem 10. *Algorithm 6 can guarantee a successful* ***channel*** *rendezvous, and the generated task allocation list $\mathbb{A}_i^t$ can satisfy the constraints in Section 3.5.7*

Proof. Since each cognitive radio of the ith SU hops following a pre-defined channel hopping sequence which can guarantee a successful channel rendezvous, our proposed scheme can also guarantee a successful channel rendezvous between two SUs as long as two of their radios achieve a channel rendezvous. Furthermore, according to **Algorithm 6**, if multiple radios hop to a same channel, all of them will wait for RTS packets. Therefore, all the constraints in Section 3.5.7 can be satisfied. $\square$

3.2.4 The Enhanced Rendezvous Scheme

One problem of the basic rendezvous scheme is that multiple radios may all use one same channel to receive RTS packets, if they all hop to the same channel simultaneously. Therefore, the cognitive radios of a SU are not well utilized. For example, when each SU is equipped with 4 cognitive radios, if 3 of them hop to channel 1 and the last radio hops to channel 2 at the same time, under the basic scheme it is equivalent to just using two radios during the current time slot. In this section, we propose an enhanced rendezvous scheme, considering multiple cognitive radios and the *send-or-receive problem*, which can well utilize all the radios of a SU during each time slot to achieve a fast *link rendezvous*.

The basic idea of our proposed enhanced rendezvous scheme is to split the total M channels into several continuous groups. Each group has exactly R consecutive channels, where R is the number of cognitive radios of a SU. If $M \bmod R \neq 0$, then sequentially add channel 1, 2, …, $R-1$ to the last group until the number of channels is R. There is an example in Fig. 3.4 when $M = 10$, $R = 4$, the generated $\lceil \frac{10}{4} \rceil = 3$ channel groups are $\{1, 2, 3, 4\}$, $\{5, 6, 7, 8\}$, and $\{9, 10, 1, 2\}$.

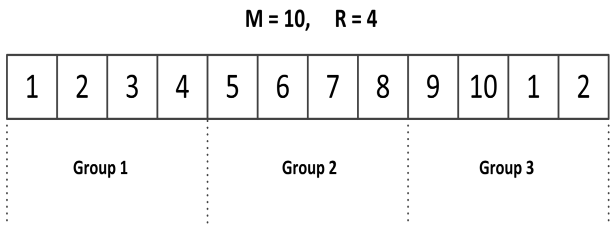


Figure 3.4: An example of generating all channel groups.

We denote $M_s = \lceil \frac{M}{R} \rceil$ as the total number of channel groups, $\mathcal{G} = \{G_1, G_2, \ldots, G_{M_s}\}$ as the set of all the channel groups, and $G_i = \{g_{i1}, g_{i2}, \ldots, g_{iR}\}$ as the channels in the ith channel group. As all SUs are equipped with exactly R cognitive radios and can access totally M channels, they can generate the same $\mathcal{G}$ independently. For a

SU, it applies M_s rather than the number of all channels M to generate a hopping sequence, based on an existing channel hopping sequence generating algorithm. We denote this sequence as the *channel-group hopping sequence* (CGHS).

Our basic idea is to assign each cognitive radio with a different current available channel during each time slot. During time slot t, if the ith SU hops to the jth channel group, it allocates current available channels in G_j sequentially to its R cognitive radios. If some channels become unavailable during the current slot which makes the total available channels in G_j are not enough for all R radios, the SU also allocates the available channels in the next channel groups until each of the R radios is allocated with a different current available channel. If the number of available channels among the total M channels is less than R, then allocate each of them to a radio and let the rest radios keep silent during the current time slot. Here is an example when $M = 10$, $R = 3$, and the available channels in each channel group are $\{2, 3\}$, $\{5\}$, $\{7, 9\}$, and $\{10, 2\}$. Assume that a SU hops to the second channel group during the current time slot according to the CGHS. The 3 allocated available channels are channel 5 from the second channel group and channel $7, 9$ from the third channel group.

The last step is to solve the *send-or-receive problem.* Note that this problem is equivalent to design a binary string of length R containing only 0 or 1. We define such a string of length R as β_R. Therefore, during time slot t, solving the *send-or-receive problem* for the ith SU is equivalent to designing a binary string β^t_{iR}. We derive the following lemmas regarding a successful *link rendezvous* between the ith and jth SU based on β^t_{iR} and β^t_{jR}.

Lemma 10. *A necessary condition of a successful **link** rendezvous between the ith and jth SU during time slot t is that β^t_{iR} XOR $\beta^t_{jR} > 0$, if we regard β^t_{iR} and β^t_{iR} as the binary representations of two integers.*

Proof. If β^t_{iR} XOR $\beta^t_{jR} = 0$, i.e., every bit of the two binary strings is the same, there does not exist a send-and-receive pair between the two SUs which can guarantee successfully sending and receiving RTS packets. However, if β^t_{iR} XOR $\beta^t_{jR} > 0$, there are at least two corresponding bits that are different and they can make one radio of a SU send and one radio of the other SU receive. $\square$

The challenge of designing β^t_{iR} is that the two SUs do not know any information about each other before a successful rendezvous. Our goal is to find a good distributed scheme to generate β so that β^t_{iR} XOR $\beta^t_{jR} = 0$ happens as less as possible. The following lemma can help us design a well performed β_R.

Lemma 11. *Two different binary strings β_{iR} and β_{jR}, always satisfy β_{iR} XOR $\beta_{jR} > 0$, as long as they are the binary representations of two different integers from the range* $[0, 2^R - 1]$.

Proof. If β_{iR} XOR $\beta_{jR} = 0$, the decimal representations of the two binary strings should be the same, as each bit of the two strings is the same, which contradicts the fact that they represent two different numbers. $\square$

The details of our proposed enhanced rendezvous scheme for the ith SU is shown in **Algorithm 7**.

Based on **Lemma 11** and **Algorithm 7**, we can get **Theorem 11**.

Theorem 11. *The enhanced rendezvous scheme in Algorithm 7 can guarantee a successful **channel** rendezvous. The probability that two SUs independently choose the same binary string for the send-or-receive problem, which may fail a link rendezvous, is $\frac{1}{2^{R\eta}}$, where R is the number of cognitive radios each SU is equipped with and η*

Algorithm 7 The enhanced blind rendezvous algorithm considering the send-or-receive problem

```
Input:
    SU's unique ID $i$
    The number of cognitive radios $R$
    The current time slot $t$
    An existing channel hopping sequence generating algorithm
 1: Get the list of channel groups $\mathcal{G}$;
 2: Based on $M_s = \lceil \frac{M}{R} \rceil$, generate a group hopping sequence $\mathbb{S}_i$;
 3: Current channel group id $k = \mathbb{S}_i^t$;
 4: for $j = 1$ to $R$ do
 5:     $c_{ij}^t$ = an available channel in $G_k, G_{k+1}, \ldots$;
 6: end for
 7: Randomly choose an integer from the range $[0, 2^R - 1]$;
 8: Get the binary representation $\beta_R$ of length $R$ of the chosen number;
 9: for $j = 1$ to $R$ do
10:     if $\beta_R[j] = 1$ then
11:         $s_{ij}^t = 1$;
12:     else
13:         $s_{ij}^t = 0$;
14:     end if
15: end for
16: for $j = 1$ to $R$ do
17:     if $s_{ij}^t = 1$ then
18:         Turn on the transceiver of the $j$th radio and send an RTS packet on channel
            $c_{ij}^t$;
19:     else
20:         Turn on the receiver of the $j$th radio and stay on channel $c_{ij}^t$ waiting for RTS
            packets;
21:     end if
22: end for
```

is the number of times that two radios of the two SUs hop to a common available channel during a blind rendezvous process.

Proof. We define a channel group between two SUs as a valid group if there is a common available channel in it. As long as two SUs have at least one common available channel, there is a valid group between them as they split all the channels into the sam channel groups. Therefore, the two SUs will hop to a valid group as long as they follow the hopping sequence which can guarantee a successful *channel rendezvous* on a common available channel. According to **Algorithm 7**, two SUs have the same send-or-receive allocations only when they choose the same random integer from the range $[0, 2^R - 1]$. The probability that two SUs independently choose the same integer from $[0, 2^R - 1]$ is $\frac{1}{2^R} \times \frac{1}{2^R} = \frac{1}{2^{2R}}$. There are totally 2^R different integers from $[0, 2^R - 1]$. Therefore, the final probability of a link rendezvous failure is $(\frac{1}{2^{2R}} \times 2^R)^\eta = \frac{1}{2^{R\eta}}$. □

3.2.5 Rendezvous Scheme with Some Information

In the last two proposed rendezvous schemes, we consider the scenario that a SU does not know any information about other SUs. However, if a SU has known the control information about some other SUs such as their locations and current available channels, the situation will be different when the SU generates the *task allocation list*. We assume that when two SUs rendezvous successfully, they exchange all of their known control information about other SUs and themselves. When considering multiple cognitive radios, several SUs may be connected together through some radios and form a connective graph (for each two SUs, there is a path for any one of them to reach the other one). After information exchange, how to optimally allocate the rest available channels to current free radios based on the obtained information about the already connected SUs to perform rendezvous with other non-connected SUs is a problem. Especially, if two SUs are located within the interference range of each other, when they allocate the same channel to their free radios, the allocation is invalid due to the interference. Therefore, when SUs are allocating channels to their free radios, they should attempt to reduce invalid channel allocations. We call this as the *channel allocation problem* during a rendezvous process when considering multiple cognitive radios.

An example is shown in Fig. 3.5. During the current time slot, SU A has rendezvoused with SU B on their commonly available channel 1 and SU C has rendezvoused with SU B on their commonly available channel 2. We assume that each SU is equipped with 3 cognitive radios, and during the current time slot, their available channel sets are $\{1, 3, 4\}$, $\{1, 2, 4\}$, and $\{1, 2, 3\}$. After information exchange, all the 3 SUs know the location and current available channel set of each other. For SU A and C, there are two remaining free cognitive radios, but SU B only has one more free radio to utilize. We assume that they are all located within the interference range of each other, which means that any two of them cannot use the same channel in order to avoid interference. Therefore, how to allocate the remaining free available channels to the remaining free cognitive radios is a problem, for instance, SU A and C cannot use channel 3 simultaneously.

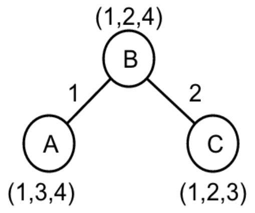


Figure 3.5: An example of the channel allocation problem.

During a time slot, we assume that there are totally N_c SUs who can get the information of each other through existing rendezvouses, and each SU is equipped with R cognitive radios. The ith SU's remaining available channel set is C_i' (some available channels have been used in establishing links such as channel 1 in the example), its location is represented as a vector L_i, and the number of remaining radios is δ_i. If a radio of a SU is allocated with an available channel and there is no other SU within its interference range using the same channel, we denote this kind of radio as a *valid allocated radio*. Our goal is to maximize the total number of *valid allocated radios* among all the N_c SUs.

We define the maximum number of SUs that can use channel i simultaneously as the *maximum utilization number* of channel i which is denoted as μ_i and the corresponding set of SUs that can use channel i simultaneously without causing any interference to each other as $\mathbb{B}_i = \{\theta_1, \theta_2, \ldots, \theta_{\mu_i}\}$, where θ_j is the ID of a SU that can use channel i. The first step is to get μ_i and $\mathbb{B}_i$ for channel i. We will solve this problem based on the following lemma.

Lemma 12. *To get μ_i and $\mathbb{B}_i$ is equivalent to solving the maximum clique problem of a corresponding undirected graph $G = \{V, E\}$.*

Proof. The maximum clique problem of an undirected graph $G = \{V, E\}$ is defined as finding a subgraph $G_m = \{V_m, E_m\} \in G$ which has the maximum number of nodes, $V_m \in V$, $E_m \in E$, and for any two different nodes $u, v \in V_m$, there is an edge $(u, v) \in E_m$ [8]. For channel i, we build graph $G = \{V, E\}$ in the following way, $V = \{1, 2, \ldots, N_c\}$, where N_c is the number of SUs who know the information of each other; an undirected edge $(u, v) \in E$ exists if SU u and SU v are not located in the interference range of each other; and $i \in C_u^{'}$, $i \in C_v^{'}$. After establishing G, the problem is equivalent to finding a subgraph $G^{'} = \{V^{'}, E^{'}\}$ of G with the maximum number of nodes and $\forall u, v \in V^{'}$, $u \neq v$, the edge $(u, v) \in E^{'}$. This is the maximum clique problem of graph G.

□

The maximum clique problem is an NP problem and there are a lot of existing algorithms to solve it [8, 9]. Here is an example of getting μ_i and $\mathbb{B}_i$ for channel i through solving the maximum clique problem. Assume that channel i is available for SU A, B, C, and D simultaneously. SU A and SU D are located within the interference range of each other. SU B and SU C are located within the interference range of each other. Therefore, according to the proof of **Lemma 12**, the established graph for this example is shown in Fig. 3.6. An edge between two SUs means that they can use channel i simultaneously since they are out of the interference range of each other. The maximum clique of this graph is very straightforward that $\mu_i = 2$ and $\mathbb{B}_i$ can be $\{A, B\}$.

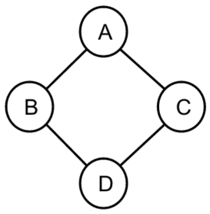


Figure 3.6: An example of the maximum clique problem.

Based on **Lemma 12**, we give the details of how to get $\mathbb{B}_i$ for channel i in **Algorithm 8**.

Based on **Algorithm 8**, we design a distributed algorithm for each SU in **Algorithm 9** which can allocate available channels to the SU's current free radios to maximize the total number of *valid allocated radios* among all N_c SUs.

Based on **Algorithm 8** and **Algorithm 9**, we can get the following theorem.

Theorem 12. *If there are N_c SUs who know the information of each other, after all the SUs execute **Algorithm 9**, the total number of valid allocated radios among*

Algorithm 8 The algorithm to get $\mathbb{B}_i$ for the ith channel

```
Input:
    Channel id: $i$
    The number of SUs who know the information of each other: $N_c$
    The set of all the $N_c$ SUs' location: $\mathcal{L} = \{L_1, L_2, \ldots, L_{N_c}\}$
    The radius of the interference range of all SUs: $I_r$
    The remaining available channel sets of all $N_c$ SUs: $C'_1, C'_2, \ldots, C'_{N_c}$
Output:
    Channel allocation set for channel $i$: $\mathbb{B}_i$
 1: Initialize a graph $G = (V, E)$ that $V = E = \phi$;
 2: $V = \{1, 2, \ldots, N_c\}$;
 3: for $u = 1$ to $N_c$ do
 4:   for $v = 1$ to $N_c$ do
 5:     if $u \neq v$, $i \in C'_u$, $i \in C'_v$, and $\| L_u - L_v \|_2 \geq I_r$ then
 6:       Add an undirected edge $(u, v)$ to $E$;
 7:     end if
 8:   end for
 9: end for
10: Solve the maximum-clique problem of $G$;
11: $\mathbb{B}_i$ = the set of nodes which forms the maximum clique of $G$;
12: return $\mathbb{B}_i$
```

all SUs will reach its maximum, and no two SUs who are located within each other's interference range use the same available channel.

Proof. If the result is not the maximum, there must be a free radio that still can be allocated with a channel without causing interference to other SUs. We assume that the jth radio of the ith SU can be allocated with channel k. We can get $\mathbb{B}_k$ from **Algorithm 8** and $\mathbb{B}_k$ is not empty. If $i \in \mathbb{B}_k$, this contradicts the fact that the jth radio of the ith SU is still free, because otherwise it should have been allocated with channel k according to **Algorithm 9**. If $i \notin \mathbb{B}_k$, there must be a radio of another SU who has been allocated with channel k. If the jth radio of the ith SU is allocated with channel k, there must be interference between the ith SU and the one already allocated with channel k. Therefore, we cannot allocate any channel to one more free radio after executing **Algorithm 9**. □

Based on **Algorithm 9**, we propose a new rendezvous scheme if each SU knows

Algorithm 9 The distributed algorithm to optimally allocate available channels for the ith SU

Input:

```
The SU's ID: $i$
Current available channel set: $C_i$
Other connected SUs' locations and available channel sets
The radius of the interference range of all SUs: $I_r$
The number of cognitive radios that have not been allocated with channels: $\delta_i$
```

Output:

```
Channel allocation set: $\mathcal{C}_i^{'}$
1: $l = \phi$;
2: **for** $j = 1$ to $M$ **do**
3:    **if** $j \in C_i$ **then**
4:       Get $\mathbb{B}_j$ through Algorithm 8;
5:       $l.append(\mathbb{B}_j)$;
6:    **end if**
7: **end for**
8: Reversely sort $l$ according to $|l[i]|$, where $|.|$ is the size of a set.
9: $\mathcal{C}_i^{'} = \phi$;
10: **for** $j = 0$ to $l.length - 1$ **do**
11:    **if** $i \in l[j]$ and $\delta_i > 0$ **then**
12:       Add the corresponding channel ID of $l[j]$ to $\mathcal{C}_i^{'}$;
13:       $\delta_i = \delta_i - 1$;
14:    **end if**
15:    **if** $\delta_i == 0$ **then**
16:       break;
17:    **end if**
18: **end for**
19: **return** $\mathcal{C}_i^{'}$
```

the control information of each other among all N_c SUs. The details are shown in **Algorithm 10**.

Algorithm 10 The rendezvous algorithm with some known information

```
Input:
    Current available channel set: $C$;
1: if a SU can get the information from other SUs then
2:     Get $\mathcal{C}'$ from Algorithm 9;
3:     Use $\mathcal{C}'$ and Algorithm 7 or Algorithm 6 to perform a rendezvous process;
4: else
5:     Use $C$ and Algorithm 7 or Algorithm 6 to perform a rendezvous process;
6: end if
```

3.3 Rendezvous Scheme Without Predetermined Send or Receiver

3.3.1 System Model And Problem Description

In this paper, we assume that there are totally N SUs and N_p PUs coexisting. There are totally M channels that all the SUs and PUs can access. Furthermore, each SU is equipped with a half-duplex cognitive radio. Therefore, a cognitive radio cannot send and receive signals simultaneously. There is no CCC in the network through which all SUs can exchange their control information. Before the initialization process of a CRN, each SU does not know any control information about other SUs, which is a very practical assumption. A time-slotted system is deployed among all SUs and PUs and they do no need to be strictly synchronized.

A channel hopping scheme is adopted for the rendezvous process between two SUs. During a rendezvous process, each SU hops to the current channel based on a channel hopping sequence, senses the current availability of the channel, and sends an RTS packet or waits for an RTS packet from another SU. Therefore, each time slot includes the sensing phase and the sending or receiving phase during the rendezvous process. According to existing rendezvous schemes, a successful rendezvous occurs when two SUs hop to a common available channel at the same time slot, but the process of sending an RTS packet and receiving a CTS reply is not considered. In this paper we define this kind of rendezvous as *channel rendezvous*. This may not be a problem when one SU is always the sender while the other is the receiver, which means that there is an explicit send-or-receive relationship between the two SUs. Under this scenario, after one SU sends an RTS packet, if the other SU receives the RTS packet and replies a CTS packet, the rendezvous succeeds.

However, during the initialization phase of a CRN, each SU may try to rendezvous with other SUs to exchange control information. There is no explicit role for a SU, since it cannot know if other SUs are senders or receivers currently. If two SUs hop to a common available channel and their transceivers are both tuning to the sending mode , none of them can receive the RTS packets, which can fail a successful rendezvous. Furthermore, each SU does not know if its target SU is on the current common available channel and is in the sending or receiving mode during the current time slot. Practically, only when the two SUs hop to a common available channel and when one SU is sending while the other is trying to receive on that channel, a successful rendezvous can finally happen. In this paper, we denote this kind of rendezvous as *link rendezvous*. Therefore, in a *link rendezvous* process without the explicit sender or receiver , a SU should not only determine the channel it should hop to but also whether tuning its half-duplex radio to the sending or receiving mode during a time slot. We call this problem as the *send-or-receive problem* in a *link rendezvous* scheme.

For the ith SU, during time slot t of a link rendezvous process, it generates a *task allocation* $\mathbb{A}_i^t = (c_i^t, s_i^t)$, where $1 \leq c_i^t \leq M$, $s_i^t = 1$ (send) or $s_i^t = 0$ (receive). We also denote C_i^t as the available channel set of the ith SU during time slot t. We define *link rendezvous* as follows.

Link Rendezvous: A successful link rendezvous between the ith SU and the jth SU during time slot t requires that (c_i^t, s_i^t) and (c_j^t, s_j^t) satisfy $c_i^t \in C_i^t$, $c_j^t \in C_j^t, c_i^t = c_j^t$, and s_i^t XOR $s_j^t = 1$.

In the following two sections, we propose two different distributed link rendezvous schemes based on two different basic ideas.

3.3.2 Basic Link Rendezvous Scheme

Our basic idea is that besides the channel hopping sequence, considering the send-or-receive problem, there is a *transmission sequence* S_i for the ith SU. S_i only contains 0 or 1 values which represent the sending mode or receiving mode, respectively, for each SU during each time slot, i.e, If $S_i[t] = 1$, during the current time slot, the SU should tune its transceiver to the sending mode, and If $S_i[t] = 0$, during the current time slot, the SU should tune its transceiver to the receiving mode. When tuning to the sending mode during a time slot, a SU can first send an RTS packet on the current channel and then tune to the receiving mode waiting for a replied CTS packet. When tuning to the receiving mode, a SU should tune the transceiver to the receiving mode, wait for RTS packets from other SUs on the current channel, and tune to the sending to reply a CTS packet if it received an RTS packet.

Since we try to design a distributed scheme for each SU, our idea for the basic link rendezvous scheme is to make each SU apply the same basic transmission sequence S_b. We assume that the length of the basic transmission sequence S_b is L and each SU circularly hops according to S_b, i,e, if the SU hops to the end of S_b, it should hop to the first element of S_b during the next time slot. For example, if the transmission sequence is $\{1, 0, 0, 1\}$, the real transmission sequence should be $\{1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1 \ldots\}$. However, since two SUs are not synchronized and do not know any control information about each other before a successful link rendezvous, two SUs may follow different real transmission sequences even though they all follow the same basic transmission sequence S_b. Therefore, for the ith SU and jth SU, the real transmission sequence of the jth SU S_j should be a circular shift from the ith SU's real transmission sequence S_i. For example, if the basic transmission sequence S_b is $\{1,0,0,1\}$, S_i is $\{1,0,0,1,1,0,0,1,\dots\}$, and S_j is $\{1,1,0,0,1,1,0,0,\dots\}$.

We define the element pair (S_i^t, S_j^t) as the *transmission pair* of the ith and jth SU. According to the previous analysis, only when one SU is on sending mode while the other is on receiving mode, two SUs can successfully finish a RTS and CTS exchange. Therefore, only the transmission pair $(0,1)$ or $(1,0)$ can result in a successful link rendezvous. We define $(0,1)$ or $(1,0)$ as the *valid transmission pair.* We define the *valid number* $N_v = \mathcal{V}(S_i, S_j)$ as the number of valid transmission pairs two real transmission sequences could get within the period L, where

$$\mathcal{V}(S_i, S_j) = \sum_{k=0}^{L-1} [(S_i[k] \ XOR \ S_j[k]) == 1]. \tag{3.4}$$

Furthermore, the average valid number $\overline{N_v}$ of a basic transmission sequence S_b is defined as

$$\overline{N_v} = \mathcal{B}(S_b) = \frac{1}{L} \sum_{k=0}^{L-1} \mathcal{V}(S_i, \mathcal{H}(S_j, k)), \tag{3.5}$$

where $\mathcal{H}(S_j, k)$ is defined as the function to make S_j perform k times circularly left shifts. A larger $\overline{N_v}$ means that there are more valid transmission pairs within a period which can make a successful link rendezvous happen with a higher probability. Therefore, our goal is to find the optimal basic transmission sequence S_b for a given length L, which can finally result in the largest average valid number $\overline{N_v}$.

To get the optimal basic transmission sequence S_b with length L, we first derive the following lemma:

Lemma 13. *For a basic transmission sequence with length L, if there are n 1 elements in it. The average valid number $\overline{N_v}$ is $2(nL - n^2)/L$.*

Proof. Considering all the L scenarios regarding different circular shifts regarding a basic transmission sequence, each 1 element can generate total $L-n$ such $(1,0)$ pairs which are valid transmission pairs. Therefore there are total $2n(L-n)$ $(1,0)$ pairs two basic transmission sequences can get. Finally, the average valid number considering all L different circular shifts is $2(nL-n^2)/L$. □

Lemma 14. *When the number of* 1 *elements in* S_b *is* $\frac{L}{2}$*, The average valid number* $\overline{N_v}$ *reaches its maximum value* $L/2$.

Proof. For a given L, $\overline{N_v} = \frac{2}{L}(n_rL - n_r^2) = \frac{2}{L}(-(n_r - \frac{L}{2})^2 + \frac{L^2}{4})$. Therefore, when $n = \frac{2}{L}$, $\overline{N_v}$ reaches the maximum value $\frac{2}{L} \times \frac{L^2}{4} = \frac{L}{2}$. □

Lemma 13 indicates that $\overline{N_v}$ is only related to the number of 1 elements in S_b regardless of their positions, for a given L. Therefore, each SU can just generate its own basic transmission sequence of length L which also should contains $\frac{L}{2}$ 1 elements. We propose a distributed algorithm for each SU to achieve a successful link rendezvous in **Algorithm 11** in which we let L equal to the number of total channels.

Algorithm 11 The basic distribute rendezvous algorithm for each SU

```
Input:
    The total number of channels: $M$
    A channel hopping sequence: $S$
 1: Randomly generate a basic transmission sequence $S_b$ with length $M$;
 2: Randomly choose a starting index $i$ on $S_b$;
 3: $count = 0$; //count time slots
 4: while Not rendezvous do
 5:     Hop to the current channel according to $S$.
 6:     if the current channel is available and $S_b[i] == 1$ then
 7:         Tune to the sending mode and send RTS packets;
 8:     end if
 9:     if the current channel is available and $S_b[i] == 0$ then
10:         Tune to the receiving mode;
11:     end if
12:     i++;
13:     if $i > M$ then
14:         i = 1; //circularly hop
15:     end if
16: end while
```

Here is an example of the basic link rendezvous scheme in Fig. 3.7, where $M = 4$, the transmission sequence of SU1 is $\{1, 0, 0, 1\}$, the transmission sequence of SU2 is $\{1, 1, 0, 0\}$, each SU hops according to its own channel hopping sequence, and the common available channels for them are channel 3 and 2. The two SUs first hop to the common available channel 3. However, they do not achieve a successful link rendezvous since they are all tuning to the sending mode during that time slot. Therefore, they just continue the channel hopping process until they all hop to the common available channel 2. Finally, they achieve a successful link rendezvous on channel 2 since SU1 is tuning to the sending mode and SU2 is tuning to the receiving mode.

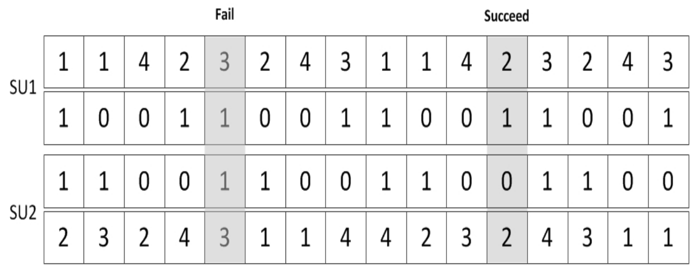


Figure 3.7: An example of the basic link rendezvous scheme.

Based on **Algorithm 11**, we can get the following theorem:

Theorem 13. *The expectation of time slots to achieve a link rendezvous for* ***Algorithm 11*** *is $E[T] = 2T_e$, where T_e is the expectation of the channel rendezvous time for a specific given channel hopping sequence.*

Proof. According to the design of the transmission sequence of each SU, the probability that there is one sender and one receiver when two SUs hop to a common available channel is 0.5, Therefore, if two SUs hop to a common available channel, the probability of a successful link rendezvous is 0.5. If a link rendezvous fails because of that there is not a valid transmission pair, two SUs should continue the channel hopping process until a successful link rendezvous. Finally, the expectation of time slots to achieve a link rendezvous time is:

$$\sum_{i=1}^{\infty} T_e(1-0.5)^{(i-1)}(0.5) = \frac{T_e}{0.5} = 2T_e. \tag{3.6}$$

$\square$

3.3.3 Enhanced Link Rendezvous Scheme

In the basic link rendezvous scheme, each SU should follow two sequences, the channel hopping sequence and the transmission sequence. In this section, we propose an enhanced link rendezvous scheme that each SU can just follow a *virtual channel hopping sequence* so each SU does not need to implement an existing channel hopping sequence and a transmission sequence simultaneously.

Our basic idea is to combine the channel hopping sequence with the transmission sequence. Therefore, we expand the total M practical channels to $2M$ *virtual channels*. For real channel i, we map it to two virtual channels i and $i+M$. An example of generating the virtual channels when M equals 4 is shown in Fig. 3.8. Therefore, after this mapping process, for each SU, there are total $2M$ virtual channels. Each SU hops according to a *virtual channel hopping sequence* generated from the total $2M$ virtual channels. When a SU hops to the virtual channel i that $i \leq M$, it should tune its transceiver to the sending mode and hop to real channel i. On the other side, when a SU hops to the virtual channel i that $M < i \leq 2M$, it should tune its transceiver to the receiving mode and hop to real channel $i-M$. Furthermore, if channel i is available to a SU during current time slot, both virtual channels i and $i+M$ are also available to the SU.

Therefore, based on the virtual channel hopping sequence and the send-or-receive problem, a successful link rendezvous is defined as follows:

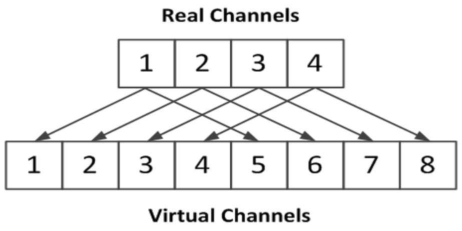


Figure 3.8: An example of generating virtual channels.

Successful link rendezvous: During time slot t, if the ith SU hops to the virtual channel c_i^t and the jth SU hops to the virtual channel c_j^t, a successful link rendezvous is achieved if $|c_i^t - c_j^t| = M$, $c_i^t \in \mathcal{C}_i^t$, and $c_j^t \in \mathcal{C}_j^t$, where $\mathcal{C}_i^t$ and $\mathcal{C}_j^t$ are the current available virtual channel sets of the ith and jth SU, respectively.

According to the definition of a successful link rendezvous based on the virtual channel hopping sequence, it is quite different from a successful *channel rendezvous* that two SUs hop to a common available channel at the same time slot. Therefore, all the existing channel hopping sequences which can guarantee a successful *channel rendezvous* cannot achieve a successful *link rendezvous* based on the virtual channels.

Our goal in this section is to design an efficient *link rendezvous* scheme which can achieve a fast successful link rendezvous between two SUs. The first step is to design a virtual channel hopping sequence based on the total $2M$ virtual channels. The basic idea about the virtual channel hopping sequence is to make each SU performs channel hopping according to the same virtual channel hopping sequence $\mathcal{S}$ generated from $2M$ virtual channels. However, since two SUs are fully distributed and not synchronized, even though they follow the same virtual channel hopping sequence, they may stay on the different positions of it. Therefore, the sequence should try to achieve a successful link rendezvous regardless of different positions of two SUs on the same virtual channel hopping sequence.

The first step to generate such a virtual channel hopping sequence is to generate a basic sequence $\mathcal{B}$ such that its length is $4M$ and $\mathcal{B}[i] = \mathcal{B}[j] = j - i, 1 \leq i < j \leq 4M$, for any $1 \leq \mathcal{B}[i] \leq 2M$, where $\mathcal{B}[i]$ is the ith element of $\mathcal{B}$. Therefore, each number in the range $[1, M]$ appears exactly twice in $\mathcal{B}$.

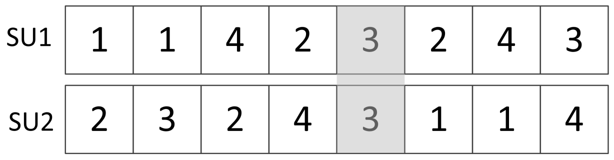
Based on the definition of $\mathcal{B}$, we can first get the following lemma regarding the existence of $\mathcal{B}$.

Lemma 15. *If two SUs follow the same basic virtual channel hopping sequence $\mathcal{B}$, the first SU starts from the ith element , the second SU starts from the jth element, and $k = |i - j| > 0$, they could both hop to virtual channel k if $k \leq 2M$, or virtual channel $4M - K$ if $2M < k < 4M$, within $4M$ time slots.*

Proof. According to the definition of $\mathcal{B}$, if $k \leq 2M$, there are two elements in the sequence $\mathcal{B}[u] = \mathcal{B}[v] = k, v > u$, and $|v - u| = k$. Therefore, the two SUs can all hop to the virtual channel k within $4M$ time slots. If $2M < k \leq 4M$, since two SUs hop according to the same $\mathcal{B}$ circularly, the distance between them on the sequence is also $4M - K < 2M$. Therefore, according to the definition of $\mathcal{B}$, there are two elements in the sequence $\mathcal{B}[u] = \mathcal{B}[v] = 4M - k, v > u$, and $|v - u| = 4M - k$.

□

Here is an example in Fig.3.9. In the example, we assume that $M = 2$, SU1 starts from the first channel in $\mathcal{B}$, and SU2 starts from the fourth channel in $\mathcal{B}$, during the fifth time slot, both SUs hop to channel 3.



SU1	1	1	4	2	3	2	4	3
SU2	2	3	2	4	3	1	1	4

Figure 3.9: An example of the basic virtual channel hopping sequence.

To get the final virtual channel hopping sequence $\mathcal{S}$, The second step is to replace some elements in $\mathcal{B}$ according to the following manipulations: for each number k

that $\mathcal{B}[i] = \mathcal{B}[j] = k, 1 \leq i < j \leq 4M$, if $k \leq M$, reset $\mathcal{B}[j] = k + M$, otherwise reset $\mathcal{B}[j] = k - M$. An example is shown in Fig. 3.10, where $M = 2$ and the basic sequence $\mathcal{B}$ could be $\{1, 1, 4, 2, 3, 2, 4, 3\}$, after the replacements, the final channel hopping sequence $\mathcal{S}$ should be $\{1, 3, 4, 2, 3, 4, 2, 1\}$.

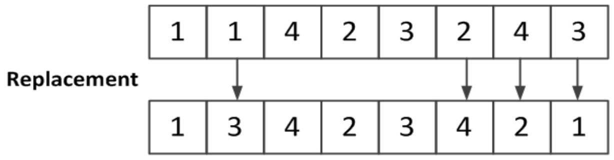


Figure 3.10: An example of the replacement manipulations.

In order to generate the virtual channel hopping sequence $\mathcal{S}$, we have the following lemmas about some important features of the basic virtual channel hopping sequence $\mathcal{B}$.

Lemma 16. *When $2M$ mod $4 = 2$, $\mathcal{B}$ does not exist.*

Proof. According to the definition of $\mathcal{B}$, designing the basic sequence is equivalent to dividing all the indexes $1, 2, \ldots, 4M$, these $4M$ numbers into two lists $\mathcal{L}_1$ and $\mathcal{L}_2$, each of which has exactly $2M$ different numbers, such that for $k = 1, 2, \ldots, 2M$, $\mathcal{L}_2[k] - \mathcal{L}_1[k] = k$. Then, we have

$$\sum_{k=1}^{2M} \mathcal{L}_2[k] - \sum_{k=1}^{2M} \mathcal{L}_1[k] = \sum_{k=1}^{2M} k = \frac{2M(2M+1)}{2}, \tag{3.7}$$

$$\sum_{k=1}^{2M} \mathcal{L}_2[k] + \sum_{k=1}^{2M} \mathcal{L}_1[k] = \sum_{k=1}^{4M} k = 2M(4M+1). \tag{3.8}$$

Combining (3.7) and (3.8), we can get

$$\sum_{k=1}^{2M} \mathcal{L}_2[k] = \frac{10M^2 + 3M}{2}. \tag{3.9}$$

When $2M \; mod \; 4 = 2$, M is an odd number that can be represented as $M = 2e+1, e \geq 0$, thus

$$\sum_{k=1}^{2M} \mathcal{L}_2[k] = \frac{10(2e+1)^2 + 3(2e+1)}{2} = 20e^2 + 23e + \frac{13}{2}. \tag{3.10}$$

the right-hand side of (3.10) is not an integer which contradicts the fact that it is a sum of $4M$ integers. □

Lemma 17. *When $M > 1$ and $2M \; mod \; 4 = 0$, $\mathcal{B}$ exists.*

Proof. When $M = 2$, sequence $\{1, 1, 4, 2, 3, 2, 4, 3\}$ satisfies the requirements of our desired basic link rendezvous sequence. When $M > 2$, the proof can be found in the second page of [4]. □

Based on the proof of **Lemma 16** and **Lemma 17**, we propose **Algorithm 12** to generate our virtual channel sequence $\mathcal{S}$.

According to **Algorithm 12**, if $M > 0$ and $2M \; mod \; 4 = 2$, the length of $\mathcal{S}$ should be larger than $4M$ in order to guarantee the existence of $\mathcal{S}$. Furthermore, we have the following lemma about $\mathcal{S}$ regarding a successful link rendezvous. After getting the virtual channel hopping sequence $\mathcal{S}$, we can get the following lemma regarding a successful link rendezvous:

Lemma 18. *Assume that when a link rendezvous process begins, the first SU is on ith element of $\mathcal{S}$, the second SU is on the jth element of $\mathcal{S}$, and $k = |i - j| > 0$. Within $4M$ time slots, two SUs can hop to two different virtual channels c_1 and c_2 such that $|c_1 - c_2| = M$.*

Proof. According to **Lemma 15**, within $4M$ time slots, two SUs can hop to the same virtual channel c on the different positions of $\mathcal{B}$. Since we perform the replacement manipulations of $\mathcal{B}$ to get $\mathcal{S}$, the manipulation can make the same two c values in $\mathcal{B}$ change to two different values c_1 and c_2 such that $|c_1 - c_2| = M$. Therefore, within $4M$ time slots, two SUs can hop to two different virtual channels c_1 and c_2 such that $|c_1 - c_2| = M$. □

Algorithm 12 The algorithm for generating the virtual channel hopping sequence

```
Input:
    The number of channels : $M$
Output:
    The desired rendezvous sequence: $\mathcal{S}$
 1: $n = 2M$;
 2: if $n\%4 == 2$ then
 3:    $n = n + 2$;
 4: end if
 5: if $n == 4$ then
 6:    $\mathcal{S} = \{1, 1, 4, 2, 3, 2, 4, 3\}$;
 7: else
 8:    Initialize $\mathcal{S}$ as a list of length $4n$;
 9:    Let $P$ be an empty list;
10:    $m = \lfloor n/4 \rfloor$;
11:    Add all pairs $(4m + r, 8m - r)$, for $r = 0, 1, \ldots, 2m - 1$, to $P$;
12:    Add pairs $(2m + 1, 6m)$ and $(2m, 4m - 1)$ to $P$;
13:    Add all pairs $(r, 4m - 1 - r)$, for $r = 1, 3, \ldots, m - 1$, to $P$;
14:    Add pair $(m, m + 1)$ to $P$;
15:    Add all pairs $(m + 2 + r, 3m - 1 - r)$, for $r = 0, 1, \ldots, m - 3$, to $P$;
16:    for $i = 1$ to $n$ do
17:       $\mathcal{S}[P[i].first] = \mathcal{S}[P[i].second] = P[i].second - P[i].first$;
18:    end for
19: end if
20: for $i = 1$ to $2n$ do
21:    if $\mathcal{S}[i] > 2M$ then
22:       $\mathcal{S}[i]\ \% = 2M$; //limit each value to $[1, 2M]$
23:    end if
24: end for
25: for $i = 1$ to $2n$ do
26:    //replacement manipulations
27:    if this the second time that $k = \mathcal{S}[i]$ appears then
28:       if $k \leq M$ then
29:          $\mathcal{S}[i] = \mathcal{S}[i] + M$;
30:       end if
31:       if $k > M$ then
32:          $\mathcal{S}[i] = \mathcal{S}[i] - M$;
33:       end if
34:    end if
35: end for
36: return $\mathcal{S}$;
```

An example of Lemma 18 is shown in Fig. 3.11, where $M = 2$, the virtual channel sequence is $\{1, 3, 4, 2, 3, 4, 2, 1\}$, SU1 starts from the first element, and SU2 starts from the third element. Within $4M = 8$ time slots, two SUs first hop to channel 2 and 4 that $4 - 2 = 2 = M$ during the fourth time slot, and then hop to channel 1 and 3 that $3 - 1 = 2 = M$ during the eighth time slot.

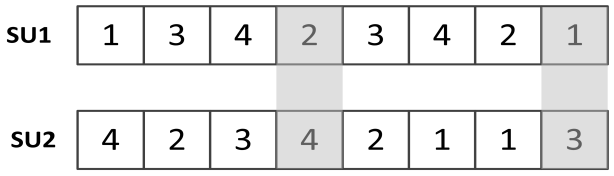


Figure 3.11: An example of the virtual channel hopping sequence.

Since a necessary condition for a successful link rendezvous is that two SUs should hop to two virtual channels c_1 and c_2, respectively, such that $c_1 > c_2, c_1 - c_2 = M$, we can design a channel hopping scheme based on **Lemma 6**. However, the problem is that if the virtual channel c_2 is not a common available virtual channel, the two SUs still cannot achieve a successful rendezvous, even if they hop to c_1 and c_2 during the same time slot. Therefore, after every $4M$ time slots, if two SUs did not achieve a successful link rendezvous, each SU should restart a new virtual channel hopping process from a new starting element in $\mathcal{S}$ to make the two SUs can hop to new virtual channels $c_1^{'}$ and $c_2^{'}$, respectively, until $c_2^{'}$ is a common available virtual channel. Therefore, we design the following distributed link rendezvous scheme for each SU in **Algorithm 13**:

Based on **Algorithm13**, we can get the following lemma:

Lemma 19. *The probability that two SUs fail to rendezvous within every $4M$ time slots is $p_f \approx 1 - p_c$, where p_c is the probability that a channel is a common available channel of the two SUs.*

Algorithm 13 The distribute link rendezvous algorithm for each SU

```
Input:
    The total number of channels: $M$
1: Generate the virtual channel hopping sequence $\mathcal{S}$;
2: Randomly choose a starting index $i$ on $\mathcal{S}$;
3: $count = 0$;// count time slots
4: while Not rendezvous do
5:     if $\mathcal{S}[i] > M$ and $\mathcal{S}[i]$ is currently available then
6:         Hop to channel $\mathcal{S}[i] - M$ and tune to the receiving mode;
7:     end if
8:     if $\mathcal{S}[i] \leq M$ and $\mathcal{S}[i]$ is currently available then
9:         Hop to channel $\mathcal{S}[i]$, tune to the sending mode, and send RTS packets on the
           channel;
10:    end if
11:    count++;
12:    if $count >= 4M$ then
13:        $count = 0$;
14:        Randomly generate a new $i$;//reset $i$
15:    else
16:        $i++$;
17:        if $i > 4M$ then
18:            $i = 1$;//circular hop
19:        end if
20:    end if
21: end while
```

Proof. The probability that two SUs choose different starting indexes in $\mathcal{S}$ is $\frac{4M(4M-1)}{(4M)^2} = \frac{4M-1}{4M}$. Two SUs can achieve a successful link rendezvous only when they choose different starting indexes and the virtual channel they will hop to is available to both of them. Therefore, the probability that two SUs achieve a successful rendezvous during every $4M$ time slots is $\frac{4M-1}{4M}p_c$. Finally, the fail probability is $1 - \frac{4M-1}{4M}p_c = \frac{4M-4Mp_c-p_c}{4M} = 1 - p_c - \frac{p_c}{4M} \approx 1 - p_c$ □

Finally, we can get the **Theorem 2** regarding the average link rendezvous time of the enhanced link rendezvous scheme in **Algorithm13**.

Theorem 14. *The expectation of link rendezvous time slots for **Algorithm13** is* $E[T] = \frac{4M}{p_c}$*, where* M *is the total number of channels and* p_c *is the probability that a channel is a common available channel of the two SUs.*

Proof. The expectation of link rendezvous time slots is:

$$\sum_{i=1}^{\infty} 4Mp_f^{(i-1)}(1-p_f) = \frac{4M}{1-p_f} = \frac{4M}{p_c}. \tag{3.11}$$

□

3.4 Rendezvous with Directional Antenna

3.4.1 System Model and Problem Description

In this paper, we consider two SUs, coexisting with several PUs, who want to rendezvous with each other. Each SU can opportunistically access a total of M channels. Each SU is equipped with a directional antenna that can transmit or receive signals on a transmission sector with a certain angle. We assume that the directional antenna of each SU can adjust the angle of a transmission sector. The two considered SUs are located within the transmission range of each other, and they can implement the same channel rendezvous scheme which can guarantee a successful channel rendezvous within a bounded time.

The transmission area of a SU is evenly divided into N sectors without overlaps. We assume that during each time slot, a SU can only transmit in one sector and on one channel. The index of each transmission sector does not change after the initialization. When two SUs with directional antennas try to set up a communication link between them, they should tune their antennas to specific directions so that their current transmission sectors can cover each other, which is called a successful *sector rendezvous*. If the SU receiver is located in the pth transmission sector of the SU sender and the SU sender is located in the qth transmission sector of the SU sender, we denote (p, q) as a *sector rendezvous pair*. We get the following theorem about the sector rendezvous and sector rendezvous pair.

Theorem 15. *There always exists a sector rendezvous between two SUs and the rendezvous pair is unique, regardless of the number of transmission sectors of each SU.*

Proof. We assume that the SU receiver is covered by the pth transmission sector of the SU sender and the SU sender is covered by the qth transmission sector of the SU receiver. Since each sector of a SU does not overlap with another sector, both p and q are unique (if a SU is located on the border of a transmission sector, only one sector is counted). Therefore, when the SU sender is on the transmission sector p and the SU receiver is on transmission sector q simultaneously, they achieve a successful sector rendezvous. Since p and q are unique, the sector rendezvous pair (p, q) is also unique. □

After tuning its directional antenna to each transmission sector, a SU performs a predetermined channel rendezvous scheme which can guarantee a successful channel rendezvous within the maximum time to rendezvous ($MTTR$). If the SU does not rendezvous with another SU within $MTTR$ time slots in the current transmission sector, it hops to the next transmission sector and repeats the channel rendezvous process.

Therefore, in this paper, we only consider how to let two SUs achieve a successful sector rendezvous, given a specific channel rendezvous scheme that can guarantee a successful channel rendezvous.

We define the process that a SU hops to a transmission sector as the *sector hopping process*. If we denote the index of the sector a SU is on at time slot t as S_t, the sequence $S_0, S_1, \ldots, S_t, \ldots$ is called a *sector hopping sequence*. An example in Fig. 3.12 shows why both SUs should perform the sector hopping process by following a sector hopping sequence. If the SU receiver just stays within a sector waiting for the RTS packet, it may never hear the SU sender since their transmission sectors may never cover each other like the scenario in Fig. 3.12. Since a SU does not know others' location before a successful blind rendezvous, both SUs should perform the sector hopping process to achieve a successful sector rendezvous.

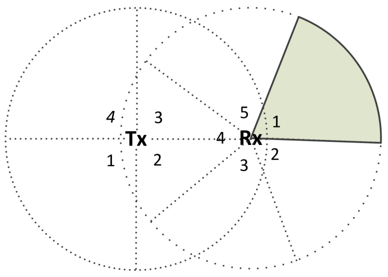


Figure 3.12: An example to show the necessity of the sector hopping process.

The *sector rendezvous problem* is defined as follows. We denote the sector hopping sequences of the SU sender and receiver as $S^s_{t_0}, S^s_{t_1}, \ldots, S^s_{t_t}$ and $S^r_{t_0}, S^r_{t_1}, \ldots, S^r_{t_t}$, respectively, where t_0 is the time the SU sender starts the rendezvous process by sending a RTS packet. Two SUs may not start the sector rendezvous process simultaneously under the blind information scenario. We assume that the rendezvous pair of the two SUs is (p, q) and they achieve sector rendezvous at time t_t if $S^s_{t_t} = p, S^r_{t_t} = q$. Our goal is to design a distributed sector hopping sequence for each SU that can guarantee a successful sector rendezvous and the time for the rendezvous $t_t - t_0$ is bounded.

The first problem about the sector rendezvous problem is the *indexing problem*. As two SUs do not know any information about each other before a successful channel rendezvous, the way they index their sectors may be quit different and unknown to each other, which may lead to $p \neq q$ for a rendezvous pair. Fig. 3.13 shows an example of the indexing problem, in which the rendezvous pair is $(2, 4)$. We denote the number of the sectors of the SU sender and SU receiver as N_s and N_r, respectively. We can notice that the indexes of all the sectors of a SU form a permutation of the numbers from 1 to N_s or N_r. Our goal is to design a distributed sector hopping sequence for each SU which can guarantee a successful sector rendezvous within a bounded time considering the indexing problem.

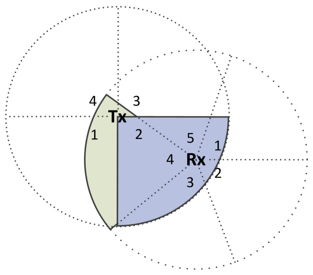


Figure 3.13: An example of the indexing problem and different sector number problem.

The second problem is called the *different sector number problem* which means that the number of sectors of each SU may be different and each SU does not know the number of sectors of the other SU before exchanging information. This is a very practical issue when two SUs' transmission parameters are different due to hardware constrains or surrounding environments. Therefore, the optimal transmission angle of each SU's directional antenna may be different. Especially, when two SUs do not know any information about each other during a blind rendezvous process, how can they configure the same transmission parameters for their antennas? An example is shown in Fig. 3.13, where $N_s = 4$ and $N_r = 5$. Under this scenario, our goal is to design a distributed sector hopping sequence for each SU which can guarantee a successful sector rendezvous within a bounded time, regardless of different sector indexes and different total number of sectors.

In the following sections, we propose distributed schemes that can guarantee a successful sector rendezvous considering both the indexing problem and different sector number problem.

3.4.2 Rendezvous Scheme Considering the Same Number of Sectors

We first propose a sector rendezvous scheme considering the same number of sectors and the indexing problem. This may not be a general scenario. However, this scheme is a basis for the scheme considering the general scenario.

Under this scenario, we denote the number of sectors for the SU sender and SU receiver as $N_s = N_r = N$ and the rendezvous sector pair is (p, q), where $1 \leq p, q \leq N$. According to the indexing problem, (p, q) could be any element from $\{1, 2, \ldots, N\} \times \{1, 2, \ldots, N\}$, where $\times$ is the Cartesian product of the two sets. Our goal is to design a sector rendezvous scheme which can guarantee a successful sector rendezvous under this scenario.

In our proposed scheme, the SU receiver always hops circularly according to the sector hopping sequence which is a circularly left shift of the sequence $1, 2, \ldots, N$. The shift is based on the starting index. The SU receiver can start sector hopping from any sector index. For instance, when $N = 5$, the sector hopping sequence is $3, 4, 5, 1, 2, 3, 4, 5, 1, 2, \ldots$ if it starts from sector 3 based on its indexing method. The SU sender performs the same way as the SU receiver but increasing the starting index by 1 after each round. For example, when $N = 5$, the sector hopping sequence for the SU sender will be $4, 5, 1, 2, 3, 5, 1, 2, 3, 4, \ldots,$ if it starts from sector 4 based on the SU sender's indexing method. If the rendezvous pair is $(1, 4)$, two SUs can achieve a sector rendezvous after 6 times of sector hopping. Fig. 3.14 shows an example of a successful sector rendezvous under the same sector number. The distributed algorithms for the

SU receiver and SU sender to generate the sector hopping sequence are shown in **Algorithm 14** and **Algorithm 15**, respectively.

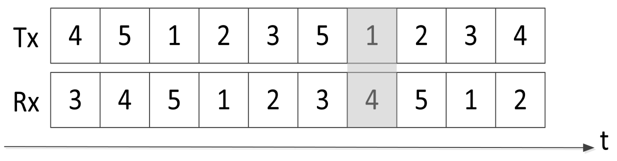


Figure 3.14: An example of a sector rendezvous under the same sector number.

Algorithm 14 Generate the sector hopping sequence for the SU receiver under the same sector number

```
Input:
    The number of sectors: $N$
    Current index of sector: $\theta_i$
Output:
    The index of the next sector: $\theta_{i+1}$
 1: if $\theta_i$ is null then
 2:     // this is the first index
 3:     $\theta_{i+1}$ = a random number from $[1, N]$;
 4: else
 5:     $\theta_{i+1} = \theta_i + 1$;
 6:     if $\theta_{i+1} > N$ then
 7:         $\theta_{i+1} = 1$;
 8:     end if
 9: end if
10: return $\theta_{i+1}$
```

Based on **Algorithm 14** and **Algorithm 15**, we can get **Theorem 16**.

Theorem 16. *By following the sector hopping sequences generated by* ***Algorithm 14*** *and* ***Algorithm 15****, two SUs can achieve a successful sector rendezvous within N^2 sector hops. The average sector hops for a successful sector rendezvous is $N^2/2$.*

Proof. Assume that the sector rendezvous pair is $(p, q) \in \{1, 2, \ldots, N\} \times \{1, 2, \ldots, N\}$, where $\times$ is the Cartesian product of the two sets, and during the first round, when the SU sender is on its pth sector, the SU receiver is on its $\beta_1 = \beta_0 \% N + 1$ sector, where $1 \leq \beta_0 \leq N$. According to **Algorithm 15**, during the next $N - 1$ rounds, when the SU sender is on its pth sector, the SU receiver should be on sectors $\beta_2 =$

Algorithm 15 Generate the sector hopping sequence for the SU sender under the same sector number

```
Input:
    The number of sectors: $N$
    Current index of sector: $\theta_i$
    Starting index of current round: $\theta_s$
Output:
    The index of the next sector: $\theta_{i+1}$
 1: if $\theta_i$ is null then
 2:     // this is the first index
 3:     $\theta_{i+1}$ = a random number from $[1, N]$;
 4:     $\theta_s = \theta_{i+1}$; // update $\theta_s$
 5: else
 6:     $\theta_{i+1} = \theta_i + 1$;
 7:     if $\theta_{i+1} > N$ then
 8:         $\theta_{i+1} = 1$;
 9:     end if
10:     if $\theta_{i+1} == \theta_s$ then
11:         $\theta_s = \theta_s + 1$; // update $\theta_s$
12:         if $\theta_s > N$ then
13:             $\theta_s = 1$;
14:         end if
15:         $\theta_{i+1} = \theta_s$; // start a new round
16:     end if
17: end if
18: return $\theta_{i+1}$
```

$(\beta_0 - 1 + N)\%N + 1, \beta_3 = (\beta_0 - 2 + N)\%N + 1, \ldots, \beta_N = (\beta_0 - (N-1) + N)\%N + 1$, respectively. During the whole N sector hopping rounds, $\beta_1, \beta_2, \ldots, \beta_N$ are N different numbers in $[1, N]$. Therefore, there always exists a number $\beta_t = q$, where $1 \leq t \leq N$. Since each sector hopping round contains N sector hops, the maximum number of hops to guarantee a sector rendezvous is N^2. Since (p, q) and the starting indexes are evenly distributed, the average sector hops for a successful sector rendezvous is $N^2/2$. □

3.4.3 Rendezvous Scheme for the General Scenario

In the last section, we consider that each SU has the same number of transmission sectors. However, in practical networks, due to the hardware heterogeneity or different surrounding environments, two SUs may have different number of transmission sectors. Under this scenario, we denote the number of sectors for the SU sender and SU receiver as N_s, N_r, respectively, where N_s and N_r could be different or the same. The corresponding rendezvous sector pair is (p, q), where $1 \leq p \leq N_s$, $1 \leq q \leq N_r$. Because of the indexing problem, (p, q) could be any element from $\{1, 2, \ldots, N_s\} \times \{1, 2, \ldots, N_r\}$, where $\times$ is the Cartesian product of the two sets. Our goal is to design a distributed sector rendezvous scheme which can guarantee a successful sector rendezvous for any $(p, q) \in \{1, 2, \ldots, N_s\} \times \{1, 2, \ldots, N_r\}$ and any $N_s > 0, N_r > 0$.

The basic idea of our proposed rendezvous scheme is to let each SU adjust the number of its sectors individually to be the smallest prime number larger than current one. After this adjustment, both N_s and N_r are prime numbers.

When $N_s \neq N_r$, both the SU sender and receiver execute **Algorithm 14** to generate their sector hopping sequence. For example, when $N_s = 5$ and $N_r = 3$, the sector hopping sequences for the SU sender and receiver could be $3, 4, 5, 1, 2, 3, 4, 5, \ldots$ and $1, 2, 3, 1, 2, 3, 1, 2, 3, \ldots$, respectively. If the rendezvous pair is $(4, 1)$, two SUs can achieve a successful sector rendezvous after 7 sector hops. The example is shown in Fig. 3.15. Based on **Lemma 15**, in **Theorem 17** shown below, we can prove that when $N_s \neq N_r$, two SUs can always achieve a successful sector rendezvous by following the sector hopping sequences generated from **Algorithm 14**.

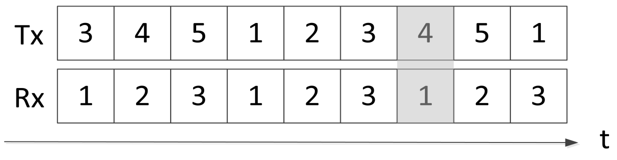


Figure 3.15: An example of a rendezvous process under different sector numbers.

Lemma 15. *If $P_1 > P_2$ are two different prime numbers, x is a random number from $[0, P_1 - 1]$, the following P_1 numbers $x\%P_1 + 1$, $(x + P_2)\%P_1 + 1$, $\ldots$, $(x + (P_1 - 1)P_2)\%P_1 + 1$ are all different.*

Proof. If there exist two different numbers $k_1 \in [0, P_1 - 1], k_2 \in [0, P_1 - 1]$ that $(x + k_1P_2)\%P_1 + 1 = (x + k_2P_2)\%P_1 + 1$. Then, $[(k1 - k2)P_2]\%P_1 = 0$. As P_2 is a prime number, P_2 and P_1 are relatively prime. Therefore, $(k1 - k2)\%P_1 = 0$. Since $0 \leq k_1 < P_1, 0 \leq k_2 < P_1$, only when $k1 = k2$, $(k1 - k2)\%P_1 = 0$, which contradicts the assumption that $k_1 \neq k_2$. □

Theorem 17. *If N_s, N_r are different prime numbers and both SUs follow the sector hopping sequence generated from **Algorithm 14**, they can achieve a successful sector rendezvous within N_sN_r sector hops.*

Proof. Without loss of generality, we assume that $N_s < N_r$, the sector rendezvous pair is (p, q), and during the first round, when the SU sender is on the pth sector, the SU receiver is on the $\beta_1 = \beta_0\%N_r + 1$ sector, where $1 \leq \beta_0 \leq N_r$. According to **Algorithm 14**, during the next $N_r - 1$ rounds, when the SU sender is on its pth sector, the SU receiver should be sequentially on $\beta_2 = (\beta_0 + N_s)\%N_r + 1, \beta_3 = (\beta_0 + 2N_s)\%N_r + 1, \ldots, \beta_{N_r} = [\beta_0 + (N_r - 1)N_s]\%N_r + 1$. According to **Lemma 15**, the

N_r numbers $\beta_1, \beta_2, \dots, \beta_{N_r}$ are all different. Therefore, there always exists a number $\beta_t = q$, where $1 \leq t \leq N_r$. Since each sector hopping round contains N_s sector hops, the maximum number of hops to guarantee a sector rendezvous is $N_s N_r$. $\square$

However, if the numbers of sectors of two SUs accidentally are the same prime number that $N_s = N_r$, the scheme we just proposed cannot work based on **Lemma 15**, while the scheme proposed in Section 3.4.2 can work. In practice, the two SUs do not know each other's sector number before a successful sector rendezvous. How can they choose the scheme they should execute individually to guarantee a successful sector rendezvous? The solution is to combine the idea in this section and the one in Section 3.4.2 together. For the SU receiver, it first adjusts the number of sectors N_r to be the smallest prime number larger than the current one. Then, it generates a sector hopping sequence based on **Algorithm 14**. For the SU sender, it adjusts the number of sectors N_s to be the smallest prime number larger than the current one. Next, it first generates a sector hopping sequence according to **Algorithm 15**. After N_s^2 sector hops, if there is not a successful sector rendezvous, it generates the sector hopping sequence based on **Algorithm 14**. Based on this idea, we propose two distributed algorithms **Algorithm 16** and **Algorithm 17** for the SU receiver and SU sender, respectively, considering both the indexing problem and the different sector number problem.

Algorithm 16 Sector rendezvous scheme for a SU receiver

```
Input:
    The number of sectors: $N_r$
1: if $N_r$ is not a prime number then
2:     Adjust $N_r$ to be the smallest prime number larger than it;
3: end if
4: Generate a sector hopping sequence based on Algorithm 14 to perform the
   sector hopping process;
```

Based on **Algorithm 16** and **Algorithm 17**, we can get **Theorem 18**.

Algorithm 17 Sector rendezvous scheme for a SU sender

```
Input:
    The number of sectors: $N_s$
1: if $N_s$ is not a prime number then
2:     Adjust $N_s$ to be the smallest prime number larger than it;
3: end if
4: Generate a sector hopping sequence based on Algorithm 15 to perform the
   sector hopping process;
5: if No successful channel rendezvous after $N_s^2$ sector hops then
6:     Generate a sector hopping sequence based on Algorithm 14 to perform the
       sector hopping process;
7: end if
```

Theorem 18. *Our proposed distributed schemes in **Algorithm 16** and **Algorithm 17** can guarantee a successful sector rendezvous within at most $N_s^2 + N_sN_r$ sector hops, considering both the indexing problem and different sector number problem.*

Proof. After adjustments, N_s and N_r should be prime numbers. If $N_s = N_r$ after the adjustments, the two SUs can achieve a successful sector rendezvous within N_s^2 sector hops according to **Theorem 16**. If $N_s \neq N_r$ after the adjustments, the two SUs may not achieve a successful sector rendezvous within N_s^2 sector hops. Therefore, after N_s^2 sector hops, the SU sender can realize that $N_s \neq N_r$ and then generates a sector hopping sequence based on **Algorithm 14**. According to **Theorem 17**, during the next N_sN_r sector hops, two SUs can achieve a successful sector rendezvous. Therefore, two SUs can achieve a successful sector rendezvous within at most $N_s^2 + N_sN_r$ sector hops, considering all scenarios. □

3.5 Power Control Protocol to Maximize The Number of Common Available Channels

3.5.1 Problem Description

Consider two SUs in a CRAHN: one is the sender and the other is the receiver during a transmission. For the SU sender, its transmission on a channel may generate direct interference to the PUs who are using the same channel. Thus, the sensing threshold of a SU sender should be set to guarantee the interference generated from it is low enough to the PUs outside the sensing range. However, for the SU receiver, it does not generate direct interference to any PU, but its reception may be interfered by a PU's transmission on the same channel. The SU receiver's sensing threshold should guarantee its received signal not be interfered seriously by PUs' transmission on the same channel. Therefore, the sensing threshold of the SU sender and receiver could be different because of their different roles in the communication. Now, the problem is: how to determine the optimal sensing threshold for both the SU sender and receiver? To solve this problem, we first show the relationship between the sensing threshold, sensing range, and transmission power for the SU sender and SU receiver in the following.

3.5.2 Analysis of the Sensing Range of a SU Sender

The requirement that a SU sender can sense whether there is an interfering transmission power on channel i is that the sensed power from channel i should be larger than a threshold $P_{th\_su\_s}$ [10]. The relationship between the sensing threshold and $P_{th\_su\_s}$ can be formulated by using the widely used transmission model [11,12] that

$$P_{th\_su\_s} = \frac{P_{pu}\kappa_1}{r_{su\_s}^{\alpha}}, \tag{3.12}$$

where P_{pu} is the transmission power of a PU, κ_1 is an antenna related constant, α is the path-loss factor, and r_{su_s} is the sensing range of the SU sender. Equation (1) shows that the sensing range is determined by the sensing threshold.

Moreover, for the SU sender, it cannot use channel i when its transmission power on channel i produces an unacceptable interference to a PU receiver. According to the definition of the sensing range, we can get

$$\frac{P_{su}\kappa_2}{r_{su\_s}^{\alpha}} \leq P_{pu\_min}, \tag{3.13}$$

where P_{su} is the transmission power of the SU sender and P_{pu_min} is the highest acceptable interference power on channel i for a PU receiver which can be obtained by the PU's minimum required decoding power and signal-to-interference ratio (SIR). From (3.13), we can get the minimum sensing range of a SU sender

$$r_{su\_s\_min} = \left(\frac{P_{su} \kappa_2}{P_{pu\_min}} \right)^{\frac{1}{\alpha}}. \tag{3.14}$$

For a SU, a larger sensing range means more unavailable channels and less available channels. Thus, we can use the minimum sensing range in (3.14) to set the threshold in (3.12) for the sensing process of a SU sender which can also guarantee an acceptable interference to PU receivers.

3.5.3 Analysis of the Sensing Range of a SU Receiver

For a SU receiver, its sensing range also has a similar relationship as in (3.12) with its detection threshold

$$P_{th\_su\_r} = \frac{P_{pu} \kappa_3}{r_{su\_r}^{\alpha}}, \tag{3.15}$$

where $P_{th\_su\_r}$ is the sensing detection threshold of the SU receiver, P_{pu} is the transmission power of a PU, and r_{su_r} is the sensing range of the SU receiver. However, a SU receiver cannot judge whether it can use channel i simply based on whether the sensed power on channel i is higher than $P_{th\_su\_r}$, because the received signal power from the SU sender may be much higher than the interference power. Accordingly, the SU receiver should additionally check if the received signal from the SU sender can meet the minimum required SIR that

$$\frac{P_{su\_rev}}{P_{th\_su\_r}} \geq SIR_{su\_min}, \tag{3.16}$$

where P_{su_rev} is the received SU power from channel i, $P_{th\_su\_r}$ is the sensing detection threshold for the SU receiver, and SIR_{su_min} is the minimum required SIR for correctly decoding. We ignore the environment noise here. P_{su_rev} can be obtained by

$$P_{su\_rev} = \frac{P_{su}\kappa_4}{d_{su}^{\alpha}}, \tag{3.17}$$

where d_{su} is the distance between the SU sender and SU receiver. From (3.15)(3.16)(3.17), we can get the minimum sensing range of the SU receiver

$$r_{su\_r\_min} = \left(\frac{P_{pu}\kappa_3 SIR_{su\_min} d_{su}^{\alpha}}{P_{su}\kappa_4} \right)^{\frac{1}{\alpha}}. \tag{3.18}$$

From a SU receiver's angle, in order to get more available channels, we can use the minimum sensing range in (3.18) as the SU receiver's sensing range to determine its sensing threshold by using (3.15).

The sensing range of a SU will directly affect the number of its individual available channels. Equation (3.14) and (3.18) indicate that when the transmission power of the SU sender P_{su} decreases, the sensing range of the SU sender will decrease but the sensing range of the SU receiver will increase, and vice versa. Thus, there is a trade-off between the SU transmission power and the sensing range of the SU pair which is also related to their individual available channels. How to determine the optimal SU transmission power which can result in the maximum number of common available channels for a SU pair in CRAHNs is an important issue that may affect the performance of SU transmissions. In the next section, we use the SU sensing ranges to estimate the expectation of the number of common available channels between two SUs and then determine the optimal SU transmission power to maximize the number of common available channels.

3.5.4 System Model

We consider two SUs in an area whose size is $S = L \times L$. One SU is the SU sender SU_1, while the other SU is the SU receiver SU_2. The set of all channels is $\mathcal{M} = \{1, 2, \ldots, M\}$ which has M channels. We assume that all the N PUs are evenly distributed in the whole area. Their active probability is β. Each PU randomly chooses a channel to transmit, and its traffic follows the exponential distribution.

3.5.5 The Expectation of the Number of Common Available Channels

In Fig. 1, the two circles represent the sensing range of SU_1 or SU_2. The combined area of their sensing ranges A_0 has three parts: A_1, A_2, and A_3, and their sizes are S_0, S_1, S_2, and S_3, respectively.

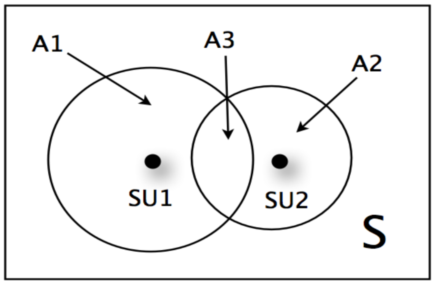


Figure 3.16: The sensing range of the SU sender and receiver.

Let $\mathcal{M}_{u\_s}$ be the set of the channels which cannot be used by the SU sender, $\mathcal{M}_{u\_r}$ be the set of channels which cannot be used by the SU receiver, and $\mathcal{M}_c$ be the set of common available channels of the SU pair. Thus, we have

$$\mathcal{M}_c = \mathcal{M} - (\mathcal{M}_{u\_s} \bigcup \mathcal{M}_{u\_r}). \tag{3.19}$$

Let $\mathcal{M}_{used} = \mathcal{M}_{u\_s} \bigcup \mathcal{M}_{u\_r}$ be the set of channels which has been used by the PUs within the sensing range of SU_1 and SU_2. Accordingly, the expectation of the number of common available channels of SU_1 and SU_2 is

$$E[|\mathcal{M}_c|] = |\mathcal{M}| - E[|\mathcal{M}_{used}|], \tag{3.20}$$

where $| \bullet |$ means the size of a set.

We use a similar method as shown in [3] to calculate $E[|\mathcal{M}_{used}|]$. Since PUs are evenly distributed in the whole area, the probability that there are k PUs in A_0 is:

$$P(n = k) = \binom{N}{k} \left(\frac{S_0}{S}\right)^k \left(1 - \frac{S_0}{S}\right)^{N-k}. \tag{3.21}$$

$$E[|\mathcal{M}_{used}|] = \sum_{k=0}^{N}\sum_{i=0}^{k}\sum_{j=0}^{i} j \binom{N}{k} \left(\frac{S_0}{S}\right)^{k} \left(1-\frac{S_0}{S}\right)^{N-k} \binom{k}{i}\beta^{i}\left(1-\beta\right)^{k-i} \frac{\binom{M}{j}S(i,j)j!}{M^{i}} \tag{3.24}$$

When the active probability of PUs is β, the probability that there are i concurrent active PUs among all the k PUs in A_0 is:

$$P(v=i|n=k) = \binom{k}{i}\beta^{i}\left(1-\beta\right)^{(k-i)}. \tag{3.22}$$

Next, we calculate the probability that there are j used channels while i active PUs in A_0. Since each PU randomly chooses a channel among all M channels with the same probability, this problem is equivalent to randomly putting i different balls into M different boxes and finding the probability of j non-empty boxes. The answer to this problem is:

$$P(u=j|v=i) = \frac{\binom{M}{j}S(i,j)j!}{M^{i}}, \tag{3.23}$$

where $S(i,j)$ is the Stirling number which represents the number of different ways to split i different elements into j different non-empty sets. From (3.21), (3.22), and (3.23), we can get the average number of the used channels as shown in (3.24) in the next page. Therefore, according to (3.20), in order to maximize the expectation of the number of common available channels, we should find the optimal P_{su} that minimizes $E[|\mathcal{M}_{used}|]$.

3.5.6 The Optimization Solution

From the above derivations, it is very challenging to get an explicit expression for the optimal P_{su}. We propose an approximation solution for the optimization problem. This solution is easy to implement in designing a practical power control protocol and can produce an improvement in the throughput according to the simulation results shown in Section **??**.

Let $E[|\mathcal{M}_{used}|]$ be a function of $\frac{S_0}{S}$, named $f(x)$.

Lemma 19. *$f(x_1) \geq f(x_2)$ when $x_1 \geq x_2$ and $N > 1$.*

Proof. We can rewrite $f(x)$ as:

$$\begin{aligned} f(x) &= \sum_{k=0}^{N} \binom{N}{k} x^k (1-x)^{N-k} g(k), \\ g(k) &= \sum_{i=0}^{k} \binom{k}{i} \beta^i (1-\beta)^{k-i} q(i), \\ q(i) &= \sum_{j=0}^{i} j p(j), \\ p(j) &= \frac{\binom{M}{j} S(i,j) j!}{M^i}. \end{aligned}$$

According to the definition of $p(j)$, we can get $\sum_{j=0}^{i} p(j) = 1$. Thus, we can induce that when $i > 1, q(i) = \sum_{j=0}^{i} jp(j) > \sum_{j=0}^{i} p(j) = 1$. Let $g(k) = \sum_{i=0}^{k} \binom{k}{i} \beta^i (1-\beta)^{k-i} t$, where t is a constant. If $t \leq 1$ then $\sum_{i=0}^{k} \binom{k}{i} \beta^i (1-\beta)^{k-i} t < \sum_{i=0}^{k} \binom{k}{i} \beta^i (1-\beta)^{k-i} q(i)$, because $q(i) > 1$ when $i > 1$. This is a paradox. Therefore, $t > 1$ when $k > 1$. Then, we can rewrite $g(k) = \sum_{i=0}^{k} \binom{k}{i} (t^{\frac{1}{i}} \beta)^i (1-\beta)^{k-i} = (t^{\frac{1}{i}} \beta + 1 - \beta)^k$. According to the definition, $0 < \beta < 1$, then we can get $g(k) > 1$ when $k > 1$. Repeat the same steps, we can also rewrite $f(x)$ as $f(x) = \sum_{k=0}^{N} \binom{N}{k} x^k (1-x)^{N-k} l$, where $l > 1$ when $N > 1$. Last, we can get the final expression of $f(x)$ as $f(x) = \sum_{k=0}^{N} \binom{N}{k} (l^{\frac{1}{k}} x)^k (1-x)^{N-k} = (l^{\frac{1}{k}} x + 1 - x)^N$. Since $l > 1$ when $N > 1$, we can finally conclude that $f(x)$ increases if x increases. □

From Lemma 1, we can know that in order to minimize $E[|\mathcal{M}_{used}|]$, we should minimize the area size S_0. Now consider the following problem: there are two circles C_0 and C_1 whose radii are r_0 and r_1. Their distance is larger than zero and does not change. Their radii only change simultaneously with a constant rate c and $-c$. Let s be the size of their combination area, then we can get the following lemma:

Lemma 20. *When $r_0 = r_1$, s will be minimum.*

Proof. Assume r_1 increases from 0 to the maximum value r_{max}. As the radius changes with a constant but opposite rate, r_0 decreases from r_{max} to 0. Let $s = G(r_0)$. We can get that $G(x)$ is symmetric according to the axis $x = \frac{1}{2}r_{max}$, since r_0 and r_1 change simultaneously with a constant but opposite rate. When r_0 decreases from r_{max} to $\frac{1}{2}r_{max}$, $G(x)$ also decreases. Because when $r_0 > r_1$ and they change simultaneously with a constant but opposite rate, the size that C_0 decreases is larger than the size that C_1 increases. Furthermore, according to the symmetry of $G(x)$, when r_0 decreases from $\frac{1}{2}r_{max}$ to 0, $G(x)$ increases. Thus, when $r_0 = \frac{1}{2}r_{max}$ and r_1 is also $\frac{1}{2}r_{max}$, $G(x)$ will reach its minimum value. □

According to Equation (3.14) and (3.18), when P_{su} increases, $r_{su\_s\_min}$ will increase and $r_{su\_r\_min}$ will decrease, and vice versa. Using Lemma 19 and Lemma 20, we can get an approximate optimal P_{su} by assuming $r_{su\_s\_min}$ and $r_{su\_r\_min}$ change with a constant but opposite rate. Therefore, letting Equation (3.14) equal (3.18), the final expression of the approximately optimal SU transmission power is

$$P_{su\_opt} = \sqrt{\frac{P_{pu\_min} P_{pu} SIR_{su\_min} d_{su}^{\alpha} \kappa_3}{\kappa_2 \kappa_4}}. \tag{3.25}$$

Fig. 3.17 shows the minimum values of $E[|\mathcal{M}_{used}|]$ when the real optimal P_{su} is used (which is obtained by the numerical method) and the results using our approximate optimal P_{su} solution, when the distance between two SUs changes from $5m$ to $15m$. We can see that the two results coincide very well, which means that our approximate optimal solution is valid and applicable.

3.5.7 The Proposed Power Control Protocol

In order to implement the optimization solution, we make the following assumptions. We assume that each SU is equipped with a GPS to get its own coordinates.

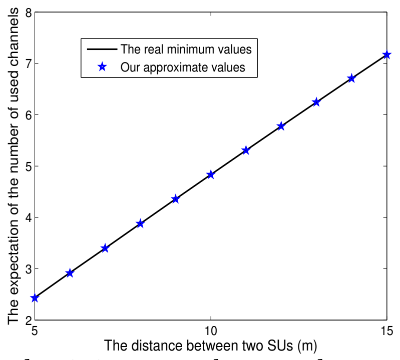


Figure 3.17: The real minimum values and our approximate values.

Moreover, according to our proposed algorithms, the SU pair should know some parameters of each other. How to get these parameters without a common control channel in CRAHNs is a challenge. In wireless ad-hoc networks, a sender and the receiver use the Request-to-Send (RTS) and Clear-to-Send (CTS) packets to inform each other before transmitting data packets. Thus, we can insert the needed information into the RTS and CTS packets. We assume that the SUs use the common hopping method in [13] to achieve successful rendezvous and the RTS and CTS packets exchange. Under this method, two SUs follow the same hopping sequence to perform channel hopping in each time slot. When one SU wants to send packets to the other, it sends the RTS packet on the currently staying channel. After receiving the CTS packet from the other SU in the later time slot, it sends data packets to the SU receiver.

In order to implement our proposed power control protocol, we design the algorithms for the SU sender and SU receiver. When the SU sender wants to send packets to the SU receiver, it will execute Algorithm 1. After receiving the RTS packet, the SU receiver will execute Algorithm 2. During the execution of the algorithms, they use the calculated optimal transmission power to get the corresponding sensing range and sensing threshold. From the pseudo code of Algorithm 1 and Algorithm 2, we can see that the complexity is $O(1)$. Thus, our algorithm has a low time cost and is easy to implement.

Algorithm 18 The algorithm for the SU sender

```
Input:
    Its coordinates $(X_{su\_s}, Y_{su\_s})$;
    $\kappa_1$, $\kappa_2$, $\alpha$, $P_{pu}$, and $P_{pu\_min}$;
Output:
    Its optimal sensing threshold $P_{th\_su\_s\_opt}$;
    Its optimal transmission power $P_{su\_opt}$;
1: Add $X_{su\_s}$ and $Y_{su\_s}$ into the RTS packet;
2: Send the RTS packet;
3: After receiving the returned CTS packet, get the optimal transmission power $P_{su\_opt}$ from the CTS packet;
4: Use $P_{su\_opt}$ and Equation (3.14) to determine its optimal sensing range $r_{su\_s\_opt}$;
5: Use $r_{su\_s\_opt}$ and Equation (3.12) to determine its optimal sensing threshold $P_{th\_su\_s\_opt}$;
6: Reset its sensing threshold as $P_{th\_su\_s\_opt}$ and transmission power as $P_{su\_opt}$;
```

Algorithm 19 The algorithm for the SU receiver

```
Input:
    $\kappa_2$, $\kappa_3$, $\kappa_4$, $\alpha$, and $P_{pu}$;
    Its minimum required SIR: $SIR_{min}$;
    The RTS packet from the SU sender;
Output:
    The SU sender's optimal transmission power $P_{su\_opt}$;
    Its optimal sensing threshold $P_{th\_su\_r\_opt}$;
1: Get $X_{su\_s}$ and $Y_{su\_s}$ from the RTS packet, and its own coordinates $X_{su\_r}$ and $Y_{su\_r}$from its GPS;
2: Use $d_{su} = \sqrt{(X_{su\_s} - X_{su\_r})^2 + (Y_{su\_s} - Y_{su\_r})^2}$ to calculate the distance to the SU sender;
3: Use (3.25) to get the optimal transmission power $P_{su\_opt}$;
4: Use $P_{su\_opt}$ and Equation (3.18) to determine its optimal sensing range $r_{su\_r\_opt}$;
5: Use $r_{su\_r\_opt}$ and Equation (3.15) to determine its optimal sensing threshold $P_{th\_su\_r\_opt}$;
6: Reset its sensing threshold as $P_{th\_su\_r\_opt}$;
7: Insert $P_{su\_opt}$ to the CTS packet and send it;
```

CHAPTER 4: Conclusion and Future Work

4.1 Conclusion of Completed Work

4.2 Future Work

4.3 Publications

REFERENCES

[1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Computer Networks (Elsevier)*, vol. 50, pp. 2127–2159, 2006.

[2] Y. Song and J. Xie, "QB^2IC: A QoS-based broadcast protocol under blind information for multi-hop cognitive radio ad hoc networks," *IEEE Trans. Vehicular Technology*, vol. 63, pp. 1453–1466, March 2014.

[3] Y. Song and J. Xie, "A QoS-based broadcast protocol for multi-hop cognitive radio ad hoc networks under blind information," in *Proc. IEEE Global Telecommunications Conference (Globecom)*, pp. 1–5, 2011.

[4] T. Skolem, "On certain distributions of integers in pairs with given differences," *Mathematica Scandinavica*, vol. 5, pp. 57–68, 1957.

[5] 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*, pp. 2444–2452, 2011.

[6] Z. Lin, H. Liu, X. Chu, and Y.-W. Leung, "Enhanced jump-stay rendezvous algorithm for cognitive radio networks," *IEEE Communications Letters*, vol. 17, September 2013.

[7] 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.

[8] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo, *The Maximum Clique Problem.* Springer, 1999.

[9] R. Carraghan and P. M. Pardalos, "An exact algorithm for the maximum clique problem," *Operations Research Letters*, vol. 9, no. 6, pp. 375–382, 1990.

[10] F. F. Digham, M.-S. Alouini, and M. K. Simon, "On the energy detection of unknown signals over fading channels," in *Proc. IEEE International Conference on Communications (ICC)*, vol. 5, pp. 3575–3579, 2003.

[11] A. Goldsmith, *Wireless Communications.* Cambridge University Press, 2005.

[12] Y. T. Hou, Y. Shi, and H. D. Sherali, "Spectrum sharing for multi-hop networking with cognitive radios," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 26, pp. 146–155, January 2008.

[13] Y. Song and J. Xie, "ProSpect: A proactive spectrum handoff framework for cognitive radio ad hoc networks without common control channel," *IEEE Trans. Mobile Computing*, vol. 11, no. 7, 2012.