# Chapter 8
# Asynchronous Blind Rendezvous Algorithms for Anonymous Users

**Abstract** In this chapter, we present *symmetric algorithms* for the blind rendezvous problem between two *asynchronous, anonymous* users. In the rendezvous setting, we fix $Alg$, $Time$, and $ID$ as follows:

$$RS =< Alg\text{-}S, Asyn, Port, Anon, Non\text{-}Obli > \tag{8.1}$$

where $Port \in \{Port - S, Port - AS\}$, which implies that we will design efficient algorithms that have good performance for both symmetric and asymmetric port situations. In this chapter, we will introduce a commonly used technique in designing rendezvous algorithms for cognitive radio networks, which is called Channel Hopping (CH) [1, 2, 11, 13, 14]. The intuitive idea is: in order to guarantee rendezvous for asynchronous users, the rule to access the licensed channels (in the network) should be periodic. Thus, we should construct a sequence of fixed length, such as $S = \{s_0, s_1, \ldots, s_{T-1}\}$ where $s_i$ is an available channel and the user hops among the channels by repeating the sequence, i.e. they access $s_{t \bmod T}$ at time $t$. Rendezvous in the distributed system is similar to rendezvous in the cognitive radio networks, and we can use the Channel Hopping technique to design efficient algorithms. In a distributed system, the available port sets for asymmetric users could be different, and different users may construct different hopping sequences. Therefore, it is difficult to design efficient algorithms (or short hopping sequences) that are suitable for all users. Moreover, the lower bound of such sequence cannot be derived directly when the sequences for different users vary, which is important for evaluating and verifying the efficiency of any proposed rendezvous algorithm. Therefore, we introduce *Global Sequence (GS) based rendezvous algorithms* to alleviate the impact of asymmetry in the ports' occupancy; the intuitive idea is: design a fixed sequence $S = \{s_0, s_1, \ldots, s_{T-1}\}$ for all users based on the full port set $U = \{1, 2, \ldots, N\}$ and each user hops among the ports by repeating the sequence (modification on the sequence may exist when some ports are not available for communication). Specifically, the GS based rendezvous algorithms work in two phases:

*Phase* 1: Assume all users have the same available port set $U$ and design the GS on the basis of $U$;



Z. Gu et al., *Rendezvous in Distributed Systems*,
DOI 10.1007/978-981-10-3680-4_8

*Phase* 2: Each user modifies the sequence according to its own available port set, i.e. when the user should access an unavailable port by the original hopping sequence, replace it with an available one that is picked randomly or by some pre-defined rules.

In this chapter, we introduce efficient GS based algorithms which can guarantee rendezvous for both symmetric and asymmetric users in a short time. To begin with, we introduce two simple algorithms for two asynchronous users that are port-symmetric in Sects. 8.1 and 8.2. Then, we introduce three GS based algorithms that have good performance for both port-symmetric and port-asymmetric scenarios. We introduce the Channel Rendezvous Sequence (CRSEQ) algorithm in Sect. 8.3, the Jump Stay (JS) algorithm in Sect. 8.4, and the Disjoint Relax Different Set (DRDS) based algorithm in Sect. 8.5. We also show the lower bound of such a GS based rendezvous algorithm in Sect. 8.6. We summarize the chapter in Sect. 8.7.

## 8.1 Generated Orthogonal Sequence (GOS)

Generated Orthogonal Sequence (GOS) [5] is considered pioneering work in cognitive radio networks, which generates a hopping sequence of length $N(N+1)$ on the basis of a random permutation of the set $\{1, 2, \ldots, N\}$. Technically, a random permutation of $\{1, 2, \ldots, N\}$ is chosen from the $N!$ permutations. Then the GOS is constructed as follows:

(1) Denote the random permutation of $\{1, 2, \ldots, N\}$ as $\{k_1, k_2, \ldots, k_N\}$;
(2) the GOS consists of $N$ phases where each phase contains $N+1$ elements;
(3) for phase $i$, construct the phase as $\{k_i, k_1, k_2, \ldots, k_N\}$;

We can regard this process as embedding the permutation $N$ times within a super-sequence of the permutation. Figure 8.1 depicts the example of the construction, where a permutation $\{2, 4, 1, 3\}$ is selected when $N = 4$, and the GOS sequence is

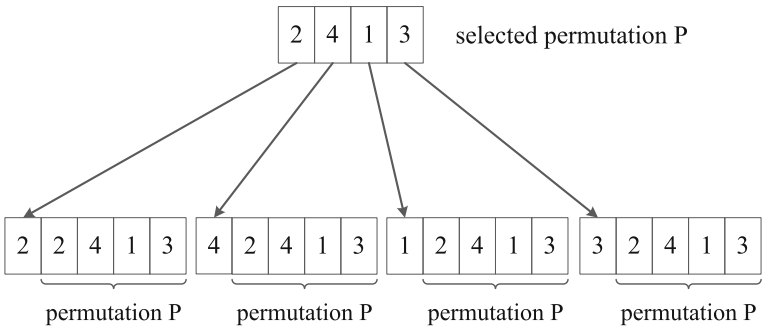


**Fig. 8.1** An example of the Generated Orthogonal Sequence

constructed by the steps introduced. However, this algorithm is limited to the situation that all channels are available. We show the correctness of GOS briefly.

First, if two users repeat the same GOS at the same time, rendezvous happens in the first time slot. So we only need to consider two asynchronous users hopping with the same GOS. Without loss of generality, assume user $u_i$ is $\delta$ time slots earlier than user $u_j$, and then there are two situations according to different $\delta$ values:

(1) $\delta\%(N+1) = 0$. It is easy to check that in the first phase of user $u_j$, it runs the same sequence of length $N$ except the first number because they all use the same permutation $\{k_1, k_2, \ldots, k_N\}$ after $k_i$ for the $i$-th phase;
(2) $\delta\%(N+1) \neq 0$. The first number of the $N$ phases of user $u_j$ will meet every number in the permutation $\{k_1, k_2, \ldots, k_N\}$ and thus rendezvous is guaranteed.

Combining these two situations, the GOS can guarantee rendezvous between two users if all channels are available.

## 8.2 Deterministic Rendezvous Sequence (DRSEQ)

GOS can guarantee rendezvous between two asynchronous users under the situation that the all ports are available (it is easy to extend the algorithm to a distributed system), which is a very special port-symmetric situation. Following this work, *Deterministic Rendezvous Sequence (DRSEQ)* of length $2N+1$ is proposed in [15], which works better under the situation that all ports are available. The main idea of the algorithm is to construct a simple sequence as:

$$\{1, 2, \ldots, N, e, N, N-1, \ldots, 1\} \tag{8.2}$$

where $e$ means the user can choose no (or any) port at this moment. Figure 8.2 shows an example of the rendezvous situations when $N = 4$. Supposing user $u_i$ starts at time 0, the constructed sequence is:

$$\{1, 2, 3, 4, e, 4, 3, 2, 1\} \tag{8.3}$$

When user $u_j$ starts the algorithm, we suppose it is $\delta$ time slots later than user $u_i$. When $\delta \in [0, 8]$, we list the rendezvous situations in the figure.

The correctness can be verified. Denote the constructed sequence as $\{a_0, a_1, \ldots, a_{2N}\}$ and each element in the sequence is constructed by the following equations:

$$a_i = \begin{cases} i+1 & \text{when } 0 \leq i < N \\ e & \text{when } i = N \\ 2N+1-i & \text{when } N+1 \leq i \leq 2N+1 \end{cases} \tag{8.4}$$

Therefore, when one user is $\delta$ time slots earlier than the other, we can use the equations to compute the rendezvous port easily. For example, when $\delta = 1$, suppose

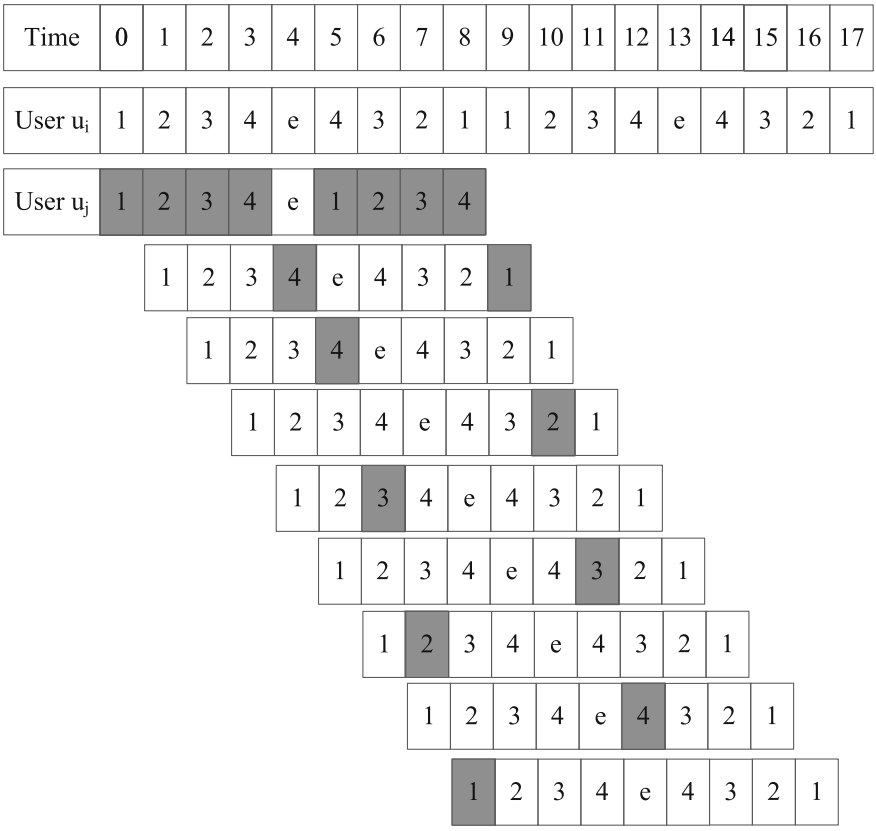


**Fig. 8.2** An example of Deterministic Rendezvous Sequence when $N = 4$

rendezvous happens at time $t$. If $0 \leq t \leq N$, we can check that $a_t \neq a_{t+1}$. If $N + 1 \leq i \leq 2N$, $a_t \neq a_{t+1}$ since $N + 1 \leq t + 1 \leq 2N + 1$. Then when $t = 2N + 1$, $a_t = 1$ and $a_{t+1} = a_1 = 1$. Therefore, rendezvous happens on port 1 and the time to rendezvous is $2N + 1$.

The algorithm can guarantee rendezvous between two asynchronous users no matter what value $\delta$ is equal to. The readers could derive the time to rendezvous of various $\delta$ values.

## 8.3 Channel Rendezvous Sequence (CRSEQ) Algorithm

Channel Rendezvous Sequence (CRSEQ) [13] is the first algorithm guaranteeing rendezvous in a bounded time when only a portion of the channels are available in a cognitive radio network. The main technique is to design a global sequence based on the triangle number. There are $P$ periods in constructing the CRSEQ, where $P$ is the smallest prime number suiting $P > N$ and each period consists of

**Table 8.1** MTTR Comparison for GS based rendezvous algorithms

| Algorithms | Symmetric | Asymmetric |
|---|---|---|
| GOS [5] | $N(N+1) = O(N^2)$ | – |
| DRSEQ [15] | $2N+1 = O(N)$ | – |
| CRSEQ [13] | $P(3P-1) = O(N^2)$ | $P(3P-1) = O(N^2)$ |
| Jump-Stay [11] | $3P = O(N)$ | $3NP(P-G) = O(N^3)$ |
| EJS [10] | $O(N)$ | $O(N^2)$ |
| DRDS [6] | $3P = O(N)$ | $3P^2+2P = O(N^2)$ |

Remarks: (1) "–" means DRSEQ and GOS are inapplicable to asymmetric users; (2) $P$ is the smallest prime number no less than $N$, $P = O(N)$

$3P-1$ number. For the $i$-th period where $i \in [1, P]$, denote the triangle number as $T_i = \frac{i(i+1)}{2}$ and the constructed period as $\{a_{i,0}, a_{i,1}, \ldots, a_{i,3P-2}\}$, the period can be constructed according to the following equations:

$$a_{i,j} = \begin{cases} T_i + j \mod P+1 & \text{when } 0 \le j < 2P-1 \\ \lfloor \frac{i}{3P-1} \rfloor \mod P+1 & \text{when } N+1 \le i \le 2N+1 \end{cases} \tag{8.5}$$

CRSEQ can guarantee rendezvous for two users in $3P^2$ time slots when the users share some common available channels. However, it works badly when the users are symmetric as shown in Table 8.1. We omit the proof of the correctness and the reader may refer to [13] for more details.

## 8.4 Jump Stay Algorithm

Jump-Stay (JS) [11] is another efficient algorithm which guarantees fast rendezvous between symmetric users in cognitive radio networks. The main idea is similar to CRSEQ, which generates the global sequence of $P$ periods and each period contains two *jump* frames and one *stay* frame (each frame contains $P$ numbers, where $P$ is the smallest prime number larger than the number of all licensed channels $N$).

We describe the two types of frames as follows. Denote the starting index as $i$ and the step length as $r$. In the jump frame, the $j$-th number (denote as $a_j$) is computed by:

$$a_j = (i + r * j - 1) \mod P+1 \tag{8.6}$$

and each frame contains $P$ numbers. In the stay frame, the users stays at channel $i$ for $P$ time slots.

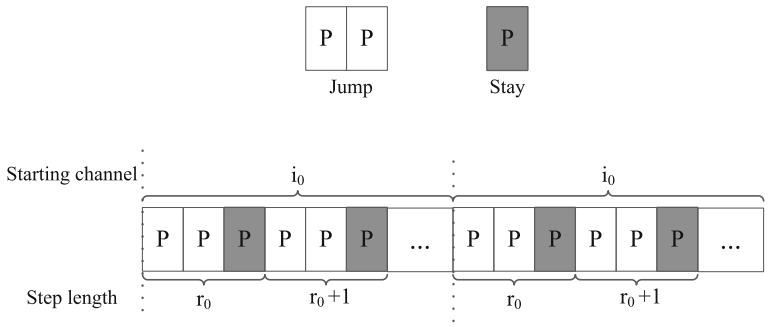


**Fig. 8.3** Jump Stay Algorithm

We describe the JS algorithm as follows:

(1) Compute $P$ as the smallest prime number larger than the number of all channels $N$;
(2) denote the initiate starting channel as $i_0 \in [1, N]$ and initiate step length $r_0 \in [1, p)$;
(3) the sequence is composed of $N$ rounds, and the starting index is $i_0 + k$ in the $k$-th round;
(4) in the first round, the starting index keeps at channel $i_0$ and there are $N$ loops inside each round;
(5) in the $i$-th loop, the step length is $r_0 + i$ and three frames are constructed as *Jump, Jump, Stay* on the basis of starting index and step length;
(6) if the chosen channel is larger than $N$, i.e. $(N, P]$, map these values to $[1, N]$ by modular operation;
(7) If the chosen channel is not available, replace it by a random available channel.

From the construction, each frame contains $P$ numbers, each loop contains 3 frames, i.e. $3P$ numbers, and each round contains $N$ loops, i.e. $3NP$ numbers. Therefore, the constructed sequence consists of $N$ rounds with $3N^2P$ numbers. Figure 8.3 illustrates the construction (in [11], $M$ represents the number of all channels). The construction can guarantee rendezvous between two (synchronous or asynchronous) (symmetric or asymmetric) users within $3N^2P$ time slots. We omit the details and reader may refer to [11] for details.

As shown in Table 8.1, although JS guarantees rendezvous between two symmetric users in a short time ($O(N)$ time slots), the *MTTR* value for two asymmetric users could be as large as $O(N^3)$ which is inacceptable.

*Enhanced Jump Stay (EJS)* [10] is a modified version such that rendezvous can be achieved in $O(N^2)$ time slots for two asymmetric users, but the main idea does not change. We would not introduce the modification and the readers who are interested in the JS algorithm could suggest some modifications that can reduce the rendezvous time.

## 8.5 Disjoint Relaxed Different Set (DRDS) Based Rendezvous Algorithm

We present an efficient rendezvous algorithm that has the best performance for both symmetric and asymmetric users [6]. The DRDS method guarantees rendezvous for two symmetric users in $O(N)$ time slots, and in $O(N^2)$ time slots for two asymmetric users. This method also can be modified such that two symmetric users can rendezvous in $O(1)$ time slots. We introduce the method in details in this section.

### *8.5.1 Global Sequence (GS)*

We define the *Global Sequence (GS)* as follow:

**Definition 8.1** We call $S = \{s_0, s_1, \ldots, s_{T-1}\}$ a Global Sequence (GS) where $\forall s_i \in S$, it is chosen from the full port set $U = \{1, 2, \ldots, N\}$.

Generally, the GS should contain every port (the label of the port) in $U$ since the users are not aware of the available ports in $U$ beforehand. We call the hopping sequence a *good GS* if the following two properties are satisfied.

**Property 8.1** *The constructed GS has a fixed length $T$.*

**Property 8.2** *For a GS $S' = \{s_0, s_1, \ldots, s_{T-1}\}$, $\forall \delta_t \geq 0$ and $\forall i \in C$, there exists $t \leq T$ such that $s_{t \bmod T} = i$ and $s_{(t+\delta_t) \bmod T} = i$.*

The first property guarantees that the users can repeat the sequence periodically since they can start the rendezvous process in different time slots. This is the main difference between two synchronous or asynchronous users. The second property guarantees rendezvous for any two port-asymmetric, asynchronous users once they begin to share some common available ports. Formally, we derive the result in the following theorem.

**Theorem 8.1** *Two (port-symmetric or port-asymmetric, synchronous or asynchronous) users can rendezvous in $T$ time slots if they both adopt a good GS of length $T$.*

*Proof* Without loss of generality, supposing the available port sets for the two users $u_i$ and $u_j$ are $C_i, C_j$ respectively, one user is $\delta \geq 0$ time slots earlier than the other; denote the good GS as $S = \{s_0, s_1, \ldots, s_{T-1}\}$ where $s_i \in U, 0 \leq i < T$.

For any common available port $k \in C_i \bigcap C_j$, according to Property 8.2, there exists $t \leq T$ such that:

$$\begin{cases} s_{t \bmod T} = k \\ s_{(t+\delta) \bmod T} = k \end{cases} \tag{8.7}$$

Since the user who starts earlier (no later than the other) accesses port $s_{t+\delta}$ while the other user accesses port $s_t$, rendezvous is achieved on port $k$ in $t \leq T$ time slots, which concludes the theorem.

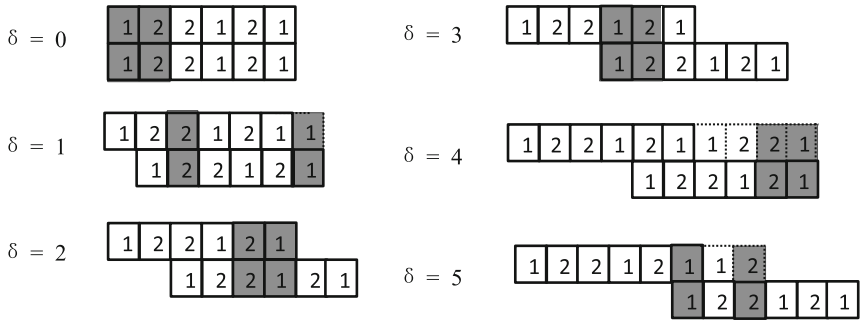


**Fig. 8.4** An example of good GS

For example, we design a good GS for two ports as $\{1, 2, 2, 1, 2, 1\}$ which is of length 6; Fig. 8.4 shows that the sequence suits Property 8.2 since rendezvous is achieved on both ports $\{1, 2\}$ when one sequence is $0 \leq \delta < 6$ time slots earlier than the other.

*Remark 8.1* Notice that any GS that guarantees rendezvous for two users should be a good GS, since the available ports are not known by the users beforehand and rendezvous has to be guaranteed on every port in $U = \{1, 2, \ldots, N\}$ no matter when the users start the rendezvous algorithm. Actually, Property 8.2 reveals the requirement to achieve rendezvous.

### 8.5.2 *Disjoint Relaxed Difference Set (DRDS)*

Before we show the method of constructing a good GS, we introduce some useful mathematical tools.

*Relaxed difference set (RDS)* is an efficient tool to construct cyclic quorum systems [8, 12]. We first introduce some definitions.

**Definition 8.2** A set $D = \{a_1, a_2, \ldots, a_k\} \subseteq Z_n$ (the set of all nonnegative integers less than $n$) is called a Relaxed Difference Set (RDS) if for every $d \neq 0 \pmod{n}$, there exists at least one ordered pair $(a_i, a_j)$ such that $a_i - a_j \equiv d \pmod{n}$, where $a_i, a_j \in D$.

RDS is a variation of the $(n, k, \lambda)$-Difference Set [4, 12] where exactly $\lambda$ ordered pairs $(a_i, a_j)$ satisfying $a_i - a_j \equiv d \pmod{n}$ are required. Given any $n$, it is proved that any difference set $D$ must have cardinality $|D| \geq \sqrt{n}$ [12]. The minimal $D$ whose size approximates the lower bound can be found when $n = k^2 + k + 1$ and $k$ is a prime power. Such a difference set is called a *Singer* Difference Set (SDS) [4]. For example, $D = \{1, 2, 4\}$ is both an SDS and an RDS under $Z_7$, but the set $D$ is an RDS, not an SDS under $Z_6$. More information about difference sets can be

found in the references and the readers may refer to them to find out more interesting properties.

In this section, we introduce one useful property which is the following.

**Lemma 8.1** *If $D$ is an RDS under $Z_n$, then $D_k = \{(a_i + k) \mod n | a_i \in D\}$ is also an RDS under $Z_n$.*

*Proof* From Definition 8.2, for every $d \neq 0 \pmod n$, there exists at least one ordered pair $(a_i, a_j)$ where $a_i, a_j \in D$ satisfy $a_i - a_j \equiv d \pmod n$. Considering the set $D_k = \{(a_i + k) \mod n | a_i \in D\}$, denote $a_{k,i} = a_i + k, \forall a_i \in D$, then $D_k = \{a_{k,1}, a_{k,2}, \ldots, a_{k,|D|}\}$. For any $d \neq 0 \pmod n$, we choose $a_{k,i}, a_{k,j} \in D_k$ such that the corresponding values $a_i, a_j \in D$ satisfy $a_i - a_j \equiv d \pmod n$; then:

$$\begin{aligned} a_{k,i} - a_{k,j} &\equiv (a_i + k) - (a_j + k) \mod n \\ &\equiv a_i - a_j \equiv d \mod n \end{aligned} \tag{8.8}$$

Therefore, the set $D_k$ is also an RDS under $Z_n$ from the definition.

Based on the definition of relaxed difference set, we introduce another important notation called *Disjoint Relaxed Difference Set*, as follows.

**Definition 8.3** A set $S = \{D_1, D_2, \ldots, D_h\}$ is called a Disjoint Relaxed Difference Set (DRDS) under $Z_n$ if $\forall D_i \in S$, $D_i$ is an RDS under $Z_n$ and $\forall D_i, D_j \in S, i \neq j$, $D_i \bigcap D_j = \emptyset$.

For example, $S = \{\{1, 2, 4\}, \{0, 3, 5\}\}$ is a DRDS under $Z_6$. Such a DRDS can be used to design GS based rendezvous algorithms and we will present the details later.

For any given integer $n$, there are many DRDSs under $Z_n$. Define Maximum DRDS $S_n$ to be the set with the largest cardinality, and it is hard to find the maximum DRDS (see Lemma 8.5 in Sect. 8.6).

### 8.5.3 Equivalence of DRDS and Good GS

Before we present the method of achieving efficient rendezvous based on the introduced notations (DRDS and good GS), we first show their equivalence.

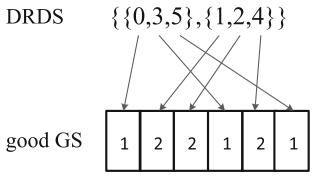


**Fig. 8.5** An example of the equivalence between good GS and DRDS

**Lemma 8.2** *Any DRDS corresponds to a* good *GS.*

*Proof* Consider a DRDS $S = \{D_0, D_1, \ldots, D_{h-1}\}$ under $Z_n$; we can construct a sequence $S' = \{s_0, s_1, \ldots, s_{n-1}\}$, as follows.

* If there exists $D_j$ such that $i \in D_j$, let $s_i = j + 1$. Otherwise, assign any value in $[1, h]$ to $s_i$.

We claim that $S'$ satisfies Properties 8.1–8.2. Obviously, Property 8.1 is satisfied since the sequence has length $n$. Then we show the satisfaction of Property 8.2. Without loss of generality, suppose one user starts $\delta_t$ time slots later. If $\delta_t \equiv 0 \pmod{n}$, Property 8.2 is satisfied apparently. Let $d' = \delta + t \bmod n$; thus $1 \leq d' < n$, and for any $i \in C$ where $C = \{1, 2, \ldots, h\}$, there exists a pair $(a_j, a_k)$ where $a_j, a_k \in D_{i-1}$ and $a_j - a_k \equiv d' \pmod{n}$. Therefore, the property suits. Combining the two aspects, $S'$ is a *good* GS.

**Lemma 8.3** *Any* good *GS corresponds to a DRDS.*

*Proof* Consider a *good* GS $S' = \{s_0, s_1, \ldots, s_{T-1}\}$ on port set $C = \{1, 2, \ldots, N\}$; we construct the DRDS $S = \{D_0, D_1, \ldots, D_{N-1}\}$ under $Z_T$ as follows:

* $D_i = \{j : s_j = i + 1, s_j \in S'\}$.

From Property 8.2, it is easy to check $S$ is a DRDS.

For example, the DRDS {{0, 3, 5}, {1, 2, 4}} corresponds to a good GS {1, 2, 2, 1, 2, 1} as Fig. 8.5.

Based on Lemmas 8.2 and 8.3, we can construct a good GS for the users to achieve rendezvous if we can design the corresponding DRDS efficiently. Moreover, if there exists some efficient method to construct such good GS, we can solve some DRDS based problems.

### *8.5.4 DRDS Construction*

To begin with, we present a DRDS construction method under $Z_n$ in linear time where $n = 3P^2$ ($P > 3$ and $P$ is a prime number).

**Algorithm 8.1** DRDS Construction of $Z_n$ when $n = 3P^2$

1: $S := \emptyset$;
2: **for** $i = 0$ to $P - 1$ **do**
3: $D_i := (Z_{(3Pi+P)} \setminus Z_{3Pi})$;
4: **for** $j = 0$ to $P - 1$ **do**
5: $q_j := j^2$, $p_{ij} := \frac{(i-q_j)(P+1)}{2} \bmod P$;
6: $t_{j0} := 3Pj + P + p_{ij}$;
7: $t_{j1} := 3Pj + 2P + p_{ij}$;
8: $D_i := D_i \bigcup \{t_{j0}, t_{j1}\}$;
9: **end for**
10: $S := S \bigcup \{D_i\}$;
11: **end for**

Algorithm 1 constructs a DRDS $S = \{D_0, D_1, \ldots, D_{P-1}\}$ as follows: divide $Z_n$ into $P$ disjoint subsets

$$Z_n = U_0 \bigcup U_1 \bigcup \cdots \bigcup U_{P-1} \tag{8.9}$$

where $U_j = Z_{3P(j+1)} \setminus Z_{3P \cdot j}$. Let

$$D_i = T_{i0} \bigcup T_{i1} \bigcup \cdots \bigcup T_{i,P-1} \tag{8.10}$$

where $T_{ij} \subseteq U_j$.

For each $U_j$, let $q_j = j^2$ and $p_{ij} = \frac{(i-q_j)(P+1)}{2} \bmod P$. Choose the $(P + p_{ij})$-th and $(2P + p_{ij})$-th number of $U_j$ to compose $T_{ij}$. They are $t_{j0}$ and $t_{j1}$ (Lines 6,7). Then $T_{ij}$ is constructed as:

$$\begin{cases} T_{ij} = \{t_{j0}, t_{j1}\} & \text{when } j \neq i \\ \{t_{j0}, t_{j1}\} \bigcup (Z_{(3Pi+P)} \setminus Z_{3Pi}) & \text{when } j = i \end{cases}$$

As the illustration in Fig. 8.6, $Z_n$ is divided into $P$ frames and each frame contains three segments of equal length. In constructing each set $D_i$, pick two numbers from the last two segments of each frame according to the above equations. In addition, all numbers in the first segment of the $i$-th frame are plugged into the set.

The intuitive idea of the construction is:

(1) In order to have some ordered pairs $(a_j, a_k)$ satisfying $a_j - a_k \equiv d \pmod{n}$ when $d$ is small from 1 to $P$, we choose the first $P$ numbers in set $U_i$, i.e. $Z_{(3Pi+P)} \setminus Z_{3Pi}$;
(2) when $d$ becomes much larger, we choose two numbers from each set $U_j$ (the last two segments of each frame) at some appropriate positions according to the modular operations in Line 5.

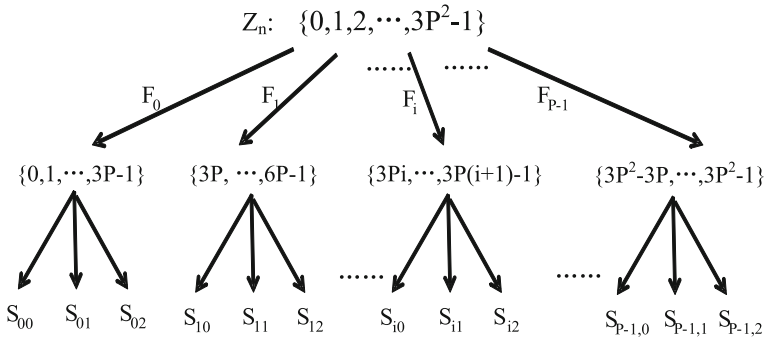


**Fig. 8.6** Illustration of DRDS construction

We present a simple example when $n = 27$:

$$D_0 = \{0, 1, 2, 3, 6, 13, 16, 22, 25\};$$
$$D_1 = \{5, 8, 9, 10, 11, 12, 15, 21, 24\};$$
$$D_2 = \{4, 7, 14, 17, 18, 19, 20, 23, 26\}.$$

It is easy to verify that $D_0$, $D_1$, $D_2$ can compose a DRDS. We prove Algorithm 1 can indeed construct a DRDS formally.

**Lemma 8.4** *Each set $D_i$ constructed in Algorithm 8.1 is a RDS.*

*Proof* From the definition of RDS, we need to prove that: for any $d \neq 0 \,(\text{mod}\, n)$, there exists at least one ordered pair $(a_j, a_k)$ satisfying $a_j - a_k \equiv d \,(\text{mod}\, n)$. Consider the following four cases:

(1) When $0 < d < P$: From Line 3 of Algorithm 1, $P$ consecutive numbers are chosen, i.e. $3Pi, 3Pi + 1, \ldots, 3Pi + P - 1 \in D_i$; thus we can find $(3Pi + d, 3Pi)$ to meet the requirement;
(2) When $P \leq d < 3P^2$ and $0 \leq d \,(\text{mod}\, 3P) < P$: Assume $d = 3Pj_1 + b_1$, $0 < j_1 < P$, $0 \leq b_1 < P$; we try to find one pair $(a_j, a_k)$ such that

$$a_j = 3Pj_2 + b_2 \mod n$$
$$a_k = 3Pj_3 + b_3 \mod n$$

where $P \leq b_3 < 2P$. If $a_j - a_k \equiv d \pmod{n}$, we can deduce $3Pj_2 + b_2 \equiv 3P(j_1 + j_3) + b_1 + b_3 \pmod{n}$, and thus $P \leq b_2 < 3P$ and both $b_2, b_3$ satisfy the equality from Lines 6, 7 of Algorithm 1. Therefore:

$$b_2 \equiv \frac{(i - j_2^2)(P + 1)}{2} \mod P$$
$$b_3 \equiv \frac{(i - j_3^2)(P + 1)}{2} \mod P$$

Thus we have:

$$\begin{cases} j_2 \equiv (j_1 + j_3) & \mod P \\ \dfrac{(i - j_2^2)(P+1)}{2} \equiv \dfrac{(i - j_3^2)(P+1)}{2} + b_1 & \mod P \end{cases}$$

Combining these equations to derive:

$$2 j_1 j_3 \equiv -(2b_1 + j_1^2) \mod P \tag{8.11}$$

Since $P$ is a prime number and $j_1, b_1$ are constant values when $d$ is fixed, $j_3$ has one unique solution in $Z_P$ [4] and we write the solution as $j^*$. Plugging $j^*$ into the above equalities, we can compute the values of $a_j$ and $a_k$.

For example, $P = 3, n = 27$, when $d = 11 = 3Pj_1 + b_1$, then $j_1 = 1, b_1 = 2$. Consider set $D_1$ and plug $j_1, b_1$ into Eq. (8.11):

$$2 j_3 \equiv -5 \equiv 1 \mod 3 \tag{8.12}$$

So $j_3 = 2$ and thus $j_2 = 0, b_3 \equiv 0 \pmod 3$. Since $3 \le b_3 < 6$, $b_3 = 3$ and then $b_2 = 5$. Therefore, $a_j = 3Pj_2 + b_2 = 5$ and $a_k = 3Pj_3 + b_3 = 21$. When $d = 11$, we can find such a pair $(5, 21)$ from $D_1$ to meet the requirement;

(3) When $P \le d < 3P^2$ and $P \le d \pmod{3P} < 2P$. Assume $d = 3Pj_1 + b_1, 0 \le j_1 < P, P \le b_1 < 2P$; let $c = \frac{(i-(i+j_1)^2)(P+1)}{2} \mod P$, $b = b_1 \mod P$ (both $c, b \in [0, P)$), and we find the pair $(a_j, a_k)$ as:

$$a_j = \begin{cases} 3P(i + j_1) + P + c & \mod n \quad \text{if } c \ge b \\ 3P(i + j_1) + 2P + c & \mod n \quad \text{if } c < b \end{cases}$$

$$a_k = \begin{cases} 3Pi + c - b & \text{if } c \ge b \\ 3Pi + P + c - b & \text{if } c < b \end{cases}$$

It can be checked that $a_j, a_k \in D_i$ and $a_j - a_k \equiv d \pmod n$.

(4) When $P \le d < 3P^2$ and $2P \le d \pmod{3P} < 3P$. Assume $d = 3Pj_1 + b_1$, $0 \le j_1 < P, 2P \le b_1 < 3P$. Find $(a_j, a_k)$ as in the second case:

$$\begin{aligned} a_j &= 3Pj_2 + b_2 \mod n \\ a_k &= 3Pj_3 + b_3 \mod n \end{aligned}$$

The difference from the second case is $2P \le b_3 < 3P$; then $P \le b_2 < 3P$ and we can find out the appropriate $j_2$, $j_3$ values. Then apply the above equalities to derive $a_j$ and $a_k$.

Based on the four cases above, $\forall d \ne 0 \pmod n$, we can find at least one ordered pair $(a_j, a_k)$ such that $a_j - a_k \equiv d \pmod n$. Therefore, each set $D_i$ constructed in the algorithm is a RDS.

Based on Lemma 8.4, we show that the constructed set of Algorithm 1 is a DRDS formally.

**Theorem 8.2** *The set* $S = \{D_0, D_1, \ldots, D_{P-1}\}$ *constructed in Algorithm 1 is a DRDS.*

*Proof* From the definition of DRDS (Definition 8.3), we prove the theorem from two aspects:

(1) Each set $D_i \in S$ is an RDS;
(2) $\forall D_i, D_j \in S, i \neq j, D_i \bigcap D_j = \emptyset$.

From Lemma 8.4, we can check that each set $D_i \in S$ is an RDS. Then, we only need to prove that $\forall D_i, D_j \in S, i \neq j, D_i \bigcap D_j = \emptyset$.

From Algorithm 1:

$$D_i = T_{i0} \bigcup T_{i1} \bigcup \cdots \bigcup T_{i,P-1}$$
$$D_j = T_{j0} \bigcup T_{j1} \bigcup \cdots \bigcup T_{j,P-1}$$

It is clear that:

$$\forall k_1 \neq k_2, T_{i,k_1} \bigcap T_{j,k_2} = \emptyset \tag{8.13}$$

Therefore, we need to show:

$$\forall 0 \leq k < P, T_{ik} \bigcap T_{jk} = \emptyset \tag{8.14}$$

There are two situations:

(1) If $k \neq i, k \neq j$, two numbers from $U_k$ are chosen for $T_{ik}$, $T_{jk}$ respectively according to $p_{ik}$ and $p_{jk}$. From Lines 6, 7 of Algorithm 1:

$$p_{ik} = \frac{(i - q_k)(P + 1)}{2} \mod P$$
$$p_{jk} = \frac{(j - q_k)(P + 1)}{2} \mod P$$

When $0 \leq i, j < P, i \neq j$ and we can conclude $p_{ik} \neq p_{jk}$. Thus $T_{ik} \bigcap T_{jk} = \emptyset$.

(2) If $k = i$ or $k = j$, the first $P$ numbers of $U_k$ will be chosen, while the other two numbers $3Pk + P + p_{ik}$ and $3Pk + 2P + p_{ik}$ do not intersect with the first $P$ numbers, and thus $T_{ik} \bigcap T_{jk} = \emptyset$.

Combining in these two situations,

$$\forall k_1, k_2, T_{i,k_1} \bigcap T_{j,k_2} = \emptyset \tag{8.15}$$

and it implies

$$D_i \bigcap D_j = \emptyset \tag{8.16}$$

Combining the two aspects, $S = \{D_0, D_1, \ldots, D_{P-1}\}$ is a DRDS.

It is obvious that Algorithm 8.1 constructs the DRDS with cardinality $\sqrt{\frac{n}{3}}$ and the algorithm runs in $O(n)$ time. The algorithm runs efficiently and it can be applied in designing efficient rendezvous algorithms.

### 8.5.5 DRDS Based Rendezvous Algorithm

Based on the DRDS construction of the special situation $n = 3P^2$ where $P$ is a prime number,[1] we present the DRDS based rendezvous algorithm as follows.

**Algorithm 8.2** DRDS Based Rendezvous Algorithm

```
1: Find the smallest prime P such that P ≥ N;
2: if P = 2 then
3:    T := 6, t := 0;
4:    S = {D0, D1}, D0 = {0, 1, 3}, D1 = {2, 4, 5};
5: else
6:    T := 3P^2, t := 0;
7:    Construct the DRDS S = {D0, D1, ..., DP−1} under ZT as Algorithm 8.1;
8: end if
9: while Not rendezvous do
10:   if 0 ≤ t < 2P then
11:      Access the port with smallest label in C′;
12:   else
13:      d := (t − 2P) mod T;
14:      Find Di ∈ S such that d ∈ Di;
15:      if Port (i + 1) ∈ C′ then
16:         Access port (i + 1);
17:      else
18:         Access an available port in C′ randomly;
19:      end if
20:   end if
21:   t := t + 1;
22: end while
```

Assume that the available port set for the user is $C' \subseteq U$ and the DRDS algorithm is described in Algorithm 8.2.

The first $2P$ time slots for the user is to access a fixed port, which resembles the listening period when a user wakes up in many asynchronous protocols; we call this the *Listening Stage*.

[1] *Bertrand-Chebyshev Theorem*: $\forall N > 1$, at least one prime $P$ exists such that $N < P < 2N$.

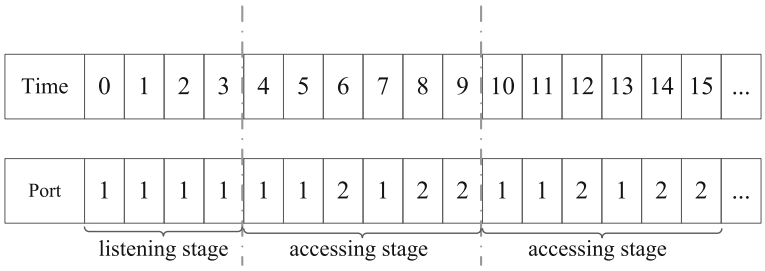


**Fig. 8.7** An example of DRDS based rendezvous algorithm (Algorithm 8.2)

Afterwards, it is the *Accessing Stage* which repeats a GS of length $T = 3P^2$ based on the DRDS construction under $Z_T$ from Algorithm 1 (if $P = 2$, the GS length is 6 and the DRDS is given as Line 2). Given any time $t$, compute $d = (t - 2P) \bmod T$ and find the RDS $D_i$ that contains $d$. The user accesses port $(i + 1)$ if it is available; otherwise, it accesses a randomly picked available port.

Figure 8.7 is an example when $N = 2$ and $C' = \{1, 2\}$. The first four time slots form the *listening stage*, and in the *accessing stage*, the user repeats the sequence $\{1, 1, 2, 1, 2, 2\}$ of length $T = 6$.

We show the correctness and efficiency of Algorithm 8.2 formally.

**Theorem 8.3** *For two users $u_i$ and $u_j$ with available port sets $C_i, C_j \subseteq U$, whenever they start Algorithm 8.2, rendezvous can be guaranteed within $MTTR = 3P = O(N)$ time slots if $C_i = C_j$, and $MTTR = 3P^2 + 2P = O(N^2)$ time slots if $C_i \neq C_j$.*

*Proof* It is easy to check that when $N \leq 2$, the theorem holds. For any $N \geq 3$, without loss of generality, suppose user $u_i$ starts earlier at time 0 and user $u_j$ starts at time $\delta_t \geq 0$. We derive the theorem from two situations:.

(1) If $C_i = C_j$, the best scenario for rendezvous is $0 \leq \delta_t < 2P$ because they are both in the *listening stage* accessing the same port. If $0 \leq (\delta_t - P) \pmod{3P} < 2P$, rendezvous occurs in the first $2P$ time slots when user $u_j$ is *listening*, while user $u_i$ is *accessing*. If $2P \leq (\delta_t - P) \pmod{3P} < 3P$, user $u_j$ can achieve rendezvous in time $[2P, 3P)$ while keeping accessing some fixed port and the $P$ numbers in $[\delta_t + 2P, \delta_t + 3P)$ for user $u_i$ are in $P$ different RDSs; so they achieve rendezvous in the *accessing stage*. Therefore, the maximum time to rendezvous is bounded in $3P$ time slots, i.e. $MTTR \leq 3P$.

(2) If $C_i \neq C_j$, we claim that rendezvous is guaranteed in $T + 2P$ time slots. Let $d = \delta_t \pmod{T}$. They may not achieve rendezvous in the *listening stage* even when $\delta_t < P$. For any common available port $i \in C_i \bigcap C_j$, we can find an ordered pair $(a_j, a_k)$ from RDS $D_{i-1}$ such that $a_j - a_k \equiv d \pmod{T}$ (Definition 8.2). So when user $u_j$'s time ticks $a_k + P$, they both access port $i$, which implies rendezvous is guaranteed within $P + a_k \leq T + 2P$ time slots.

Combining the two aspects, the theorem holds.

### 8.5.6 Improved DRDS Based Rendezvous Algorithm

Although the DRDS based rendezvous algorithm (Algorithm 2) guarantees fast rendezvous for both symmetric and asymmetric users, we can improve it such that rendezvous can be achieved in $O(1)$ time slots for two symmetric users, which matches the state-of-the-art result [3] for the cognitive radio network.

**Algorithm 8.3** Improved DRDS Based Rendezvous Algorithm

1: Find the smallest prime $P$ such that $P \geq N$;
2: Denote the port with smallest label in $C'$ as $c_m$ and the label as $m$;
3: **if** $P = 2$ **then**
4: $T_1 := 6, t := 0$;
5: $S = \{D_0, D_1\}, D_0 = \{0, 1, 3\}, D_1 = \{2, 4, 5\}$;
6: **else**
7: $T_1 := 3P^2, t := 0$;
8: Construct the DRDS $S = \{D_0, D_1, \ldots, D_{P-1}\}$ under $Z_T$ as Algorithm 8.1;
9: **end if**
10: $T := 6T_1$;
11: **while** Not rendezvous **do**
12: $f := \lfloor t/6 \rfloor, d := t\%6$;
13: **if** $d = 0$ or 1 or 3 **then**
14: Find $D_i \in S$ such that $f \in D_i$;
15: **else**
16: $i := m - 1$;
17: **end if**
18: **if** Port $(i + 1) \in C'$ **then**
19: Access port $(i + 1)$;
20: **else**
21: Access an available port in $C'$ randomly;
22: **end if**
23: $t := t + 1$;
24: **end while**

Similar to Algorithm 8.2, assuming the available port set for the user is $C'$ and denote the port with the smallest label as $c_m$ where $m$ is the label. Then Algorithm 8.3 constructs a DRDS $S_1$ similar to Algorithm 8.2. In order to guarantee fast rendezvous for two symmetric users, we expand each time slot into 6 slots where the 0, 1, 3-th numbers are the corresponding RDS in $S$ (as Line 14) while the 2, 4, 5-th numbers are the smallest labels among all available ports. The time division method is introduced in Chap. 7. Figure 8.8 shows a simple example when there are only two available ports (we use $m$ to be the smallest label in the figure other than 1).

We show the correctness and efficiency of Algorithm 8.3 as follows.

**Theorem 8.4** *For two users $u_i$ and $u_j$ with available port sets $C_i, C_j \subseteq U$, whenever they start Algorithm 8.3, rendezvous can be guaranteed within $MTTR = 6 = O(1)$ time slots if $C_i = C_j$, and $MTTR = 18P^2 = O(N^2)$ time slots if $C_i \neq C_j$.*

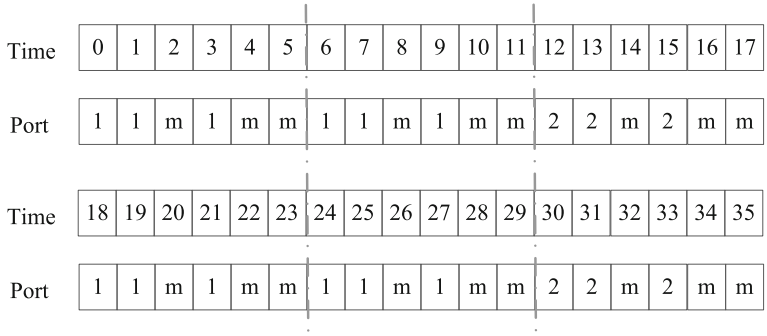


**Fig. 8.8** An example of Improved DRDS based rendezvous algorithm (Algorithm 8.3)

*Proof* It is easy to check that when $N \leq 2$, the theorem holds. For any $N \geq 3$, without loss of generality, suppose user $u_i$ starts earlier at time 0 and user $u_j$ starts at time $\delta \geq 0$. Denote the smallest labels for both users as $m_A, m_B$ respectively and we prove the theorem from two situations:

(1) If $C_i = C_j$, the smallest labels of the users are the same, i.e. $m_A = m_B$. Denote $x = \delta \% 6$ and there are six situations respectively.

   (1) If $x = 0$, both users access the port with the smallest label $m_A, m_B$ at time $\delta + 2$, thus $TTR = 3$;
   (2) if $x = 1$, user $u_i$ accesses port $m_A$ at time $\delta + 4$ while user $u_j$ accesses port $m_B$, thus $TTR = 5$;
   (3) if $x = 2$, user $u_i$ accesses port $m_A$ at time $\delta + 2$ while user $u_j$ accesses port $m_B$, thus $TTR = 3$;
   (4) if $x = 3$, user $u_i$ accesses port $m_A$ at time $\delta + 2$ while user $u_j$ accesses port $m_B$, thus $TTR = 3$;
   (5) if $x = 4$, user $u_i$ accesses port $m_A$ at time $\delta + 4$ while user $u_j$ accesses port $m_B$, thus $TTR = 5$;
   (6) if $x = 5$, user $u_i$ accesses port $m_A$ at time $\delta + 5$ while user $u_j$ accesses port $m_B$, thus $TTR = 6$;

   Thus rendezvous can be guaranteed in $MTTR = 6$ time slots.

   Actually, $\{\{0, 1, 3\}, \{2, 4, 5\}\}$ is a DRDS and the port with the smallest label occurs at the 2, 4, 5-th time slots when we expand each time slot of the GS into 6 time slots, and these positions correspond the the second RDS. Therefore, $\forall x \in [0, 5]$, the two users' hopping sequences should intersect at some positions (of the expanded 6 time slots) so that both users access the port with the smallest label, and rendezvous can be guaranteed quickly;

(2) If $C_i \neq C_j$, similar to the first situation, we can conclude that two users' hopping sequences should intersect at some positions (of the expanded 6 time slots) so that both users access the ports corresponding to the RDS in $S$ as in Line 14. From the proof of Theorem 8.3, $MTTR \leq 6 * 3P^2$ time slots.

Combining the two aspects, the theorem can be concluded.

*Remark 8.2* In Chaps. 6 and 7, we present simple algorithms for two port-symmetric users if they are aware of the symmetric situation. However, they actually cannot know whether they are symmetric or not. The method introduced in this section is a good extension by transforming any rendezvous algorithm to a new one which can guarantee rendezvous in $O(1)$ time slots, if the two users are indeed port-symmetric. In addition, it would not degrade the time complexity as compared to the original algorithm if they are port-asymmetric.

Although the improved DRDS method can guarantee rendezvous for two symmetric users in a very short time, it increases the time to rendezvous for two asymmetric users (by 6 times). In practical situations, the original DRDS algorithm (Algorithm 8.2) is more preferable.

## 8.6 Lower Bound for GS Based Rendezvous Algorithms

In order to show the efficiency of the DRDS based rendezvous algorithm, we derive a lower bound for any Global Sequence (GS) based algorithm for two users. In other words, we should find the smallest length of any good GS based on $N$ external ports, and any GS that guarantees rendezvous is a good GS (see Remark 8.1).

Since any good GS corresponds to a DRDS, one intuitive method is to find the DRDS with maximum cardinality under $Z_n$ (denote the corresponding DRDS as maximum DRDS).

**Lemma 8.5** *Given $n$, the cardinality of the maximum DRDS under $Z_n$ is bounded by $|S_n| \leq \sqrt{n}$.*

This lemma is derived easily from the fact that any RDS $D$ should have cardinality $|D| \geq \sqrt{n}$ [12].

Actually, if there exists some algorithm $\mathscr{F}$ that can compute the DRDS of maximum cardinality under $Z_n$ for any given $n > 0$, we can come up with an algorithm to derive the smallest length of good GS based on $N$ channels as follows:

1: Invoke $\mathscr{F}$ to compute the maximum cardinality of any DRDS under $Z_n$ where $n \in [N^2, 3P^2]$, where $P \geq N$ is a prime number;
2: Find the smallest $n$ in the range such that the maximum cardinality is no less than $N$.

Here the smallest $n$ is the smallest length of a good GS, i.e. the lower bound. In the first step, we try to compute the maximum cardinality of any DRDS under $Z_n$ when $n$ ranges from $N^2$ to $3P^2$. The value $N^2$ comes from Lemma 8.5, and the second value $3P^2$ comes from the DRDS based algorithm in this chapter. Therefore, the lower bound of the good GS can be derived precisely if we can compute the DRDS

**Table 8.2** Relationship between $n$ and maximum DRDS $|S_n|$ when $2 \leq n \leq 50$

| The number: $n$ | Maximum DRDS: $\lvert S_n \rvert$ |
|---|---|
| $2 \leq n \leq 5$ | 1 |
| $6 \leq n \leq 14$ | 2 |
| $15 \leq n \leq 23$ | 3 |
| $24 \leq n \leq 30$ | 4 |
| 31 | 5 |
| $32 \leq n \leq 34$ | 4 |
| $35 \leq n \leq 47$ | 5 |
| $48 \leq n \leq 50$ | 6 |

with maximum cardinality. However, it is hard to compute the maximum DRDS for any given $n$, i.e. it is hard to find the tight bound (Lemma 8.5 is a loose bound).

Actually, for any set $\mathscr{D} = \{D_0, D_1, \ldots, D_h\}$ where $D_i$ is an RDS under $Z_n$ and $h \geq \sqrt{n}$, it is hard to compute the maximum DRDS from $\mathscr{D}$ since it can be reduced from the Set Packing Problem[2] which is NP-complete [9]. When each set $|D_i| \geq \sqrt{n}$, it is equivalent to Maximum $\sqrt{n}$-Set Packing which cannot be efficiently approximated within a factor of $\Omega(\frac{\sqrt{n}}{\ln \sqrt{n}})$ [7].

We compute all RDSs with cardinality in $[\sqrt{n}, \sqrt{3n}]$ and use exhaustive search to find the maximum DRDS when $n = 2, 3, \ldots, 50$. The relationship between $n$ and the maximum DRDS (denoted as $|S_n|$) is listed in Table 8.2.

Since it is hard to compute the exact lower bound of any good GS, we try then derive a (loose) lower bound for any good GS based on the equivalence of DRDS and good GS. We first introduce an important lemma.

**Lemma 8.6** *Suppose $D$ is an RDS under $Z_T$ where $T = N(N+1)$ and $|D| = N+1$, then $N \leq 3$.*

*Proof* Consider all pairs $(a_j, a_k)$ where $a_j, a_k \in D$, $j \neq k$, and define $d_{jk} = (a_j - a_k) \bmod T$ which we call a *difference value*. $\forall d \in \{1, 2, \ldots, T-1\}$, there exists at least one difference value $d_{jk} = d$. Since there are $N(N+1)$ difference values, there exist two pairs $(a_j, a_k)$ and $(a'_j, a'_k)$ such that $d_{jk} = d_{j'k'}$ and the other difference values are all distinct. However,

$$d_{kj} = T - d_{jk} = T - d_{j'k'} = d_{k'j'} \tag{8.17}$$

which implies there exists another two pairs $(a_k, a_j)$, $(a'_k, a'_j)$ sharing a common difference value. The situation can happen only when $a_j = a'_k, a_k = a'_j$. Then

$$a_j - a_k \equiv a_k - a_j \mod T \tag{8.18}$$

and it means

[2]Given a finite set $U$ and a list of subsets of $U$, the problem asks if some $k$ subsets in the list are pairwise disjoint.

$$a_j - a_k \equiv \frac{T}{2} \mod T \tag{8.19}$$

By Lemma 8.1, construct another RDS

$$D' = \{(a - a_j) \mod T | a \in D\} \tag{8.20}$$

and thus 0, $\frac{T}{2} \in D'$.

Denote

$$S_1 = \left\{0 < a < \frac{T}{2} | a \in D'\right\}$$
$$S_2 = \left\{\frac{T}{2} < a < T | a \in D'\right\}$$

and let $d_1 = |S_1|$ and $d_2 = |S_2|$. Thus $d_1 + d_2 = |D| - 2 = N - 1$.

We count the number in set

$$S_3 = \left\{0 < a < \frac{T}{2} | a \notin D'\right\} \tag{8.21}$$

from two sides. First, since

$$|S_1 \bigcup S_3| = \frac{T}{2} - 1 \tag{8.22}$$

It is easy to compute:

$$d_1 + |S_3| = \frac{T}{2} - 1 \Rightarrow |S_3| = \frac{T}{2} - 1 - d_1 \tag{8.23}$$

From the analysis above, all other pairs satisfy:

$$(a_j, a_k) \neq \left(0, \frac{T}{2}\right) \text{ or } \left(\frac{T}{2}, 0\right) \tag{8.24}$$

and hence it should have a distinct difference value. We construct $S_3$ as follows:

(1) $\forall a \in S_1$, let $a' = \frac{T}{2} - a \in S_3$; otherwise $(a, 0)$ and $(\frac{T}{2}, a')$ share the same difference value;
(2) $\forall a \in S_2$, let $T - a \in S_3$ and $a - \frac{T}{2} \in S_3$;
(3) $\forall a_1 < a_2 \in S_1$, define $\delta = a_2 - a_1$, and let $\frac{T}{2} - \delta \in S_3$ and $\delta \in S_3$; otherwise we can find two pairs sharing a common difference value;
(4) $\forall a_1 < a_2 \in S_2$, define $\delta = a_2 - a_1, 0 < \delta < \frac{T}{2}$, and then let $\frac{T}{2} - \delta$ and $\delta$ belong to $S_3$.
(5) $\forall a_1 \in S_1, \forall a_2 \in S_2$, define $\delta = a_2 - a_1$, if $\delta > \frac{T}{2}$; rewrite $\delta = T - \delta$, and then let $\delta \in S_3$ and $(\frac{T}{2} - \delta) \in S_3$.

It is easy to verify that when we choose one value $a$ or two values $\{a, \frac{T}{2} - a\}$ to compose $S_3$, they cannot belong to $S_3$ before the step. (If $a = \frac{T}{2} - a$, we only add the value once and this special situation happens at most once.) Thus:

$$|S_3| \geq d_1 + 2d_2 + 2 \cdot \frac{d_1(d_1 - 1)}{2} + 2 \cdot \frac{d_2(d_2 - 1)}{2} + 2d_1d_2 - 1 \tag{8.25}$$

So:

$$\frac{T}{2} - 1 - d_1 \geq (d_1 + d_2)^2 + d_2 - 1 \tag{8.26}$$

Plugging $d_1 + d_2 = N - 1$, we derive:

$$N^2 \leq 3N \Rightarrow N \leq 3 \tag{8.27}$$

Therefore, the lemma holds.

Then, we derive a lower bound (not tight) for any good GS in Theorem 8.5.

**Theorem 8.5** *Any* good *GS* $S' = \{s_0, s_1, \ldots, s_{T-1}\}$ *based on* $N$ *channels satisfies:*

$$\begin{cases} T \geq N^2 + N & If\ N \leq 2 \\ N^2 + N + 1 & If\ N \geq 3\ and\ N\ is\ a\ prime\ power \\ N^2 + 2N & Otherwise \end{cases}$$

*Proof* When $N = 1$, it is clear that $T \geq 2$. Suppose $N \geq 2$; by Lemma 8.3, we can construct a DRDS as:

$$S = \{D_0, D_1, \ldots, D_{N-1}\} \tag{8.28}$$

under $Z_T$. By Lemma 8.5, we have:

$$N \leq \sqrt{T} \Rightarrow T \geq N^2 \tag{8.29}$$

Let $h = \min_{D_i \in S} |D_i|$; if $h \leq N$, the set $D_i$ (where $|D_i| = h$) has exactly $h(h-1)$ ordered pairs $(a_j, a_k)$, which implies at most $h(h-1) \leq N(N-1)$ difference values for $d$ exist such that

$$a_j - a_k \equiv d \mod T \tag{8.30}$$

When $N \geq 2$, we have:

$$N(N-1) < N^2 - 1 \leq T - 1 \tag{8.31}$$

and $D_i$ cannot be an RDS. Thus $h \geq N + 1$.

Assume $h = N + 1$, since

$$D_0 \bigcup D_1 \bigcup \cdots \bigcup D_{N-1} \subseteq Z_T \tag{8.32}$$

we derive:

$$T \geq \sum_{i=0}^{N-1} |D_i| \geq Nh = N(N+1) \tag{8.33}$$

There are three cases to be analyzed.

*Case* 1: If $T = N(N+1)$, by Lemma 8.6, $N \leq 3$. When $N = 2$, $\{\{0, 1, 3\}, \{2, 4, 5\}\}$ is a DRDS under $Z_6$. However, when $N = 3$, we cannot find a DRDS with three disjoint RDS through exhaustive search;
*Case* 2: If $T = N^2 + N + 1$, suppose $D_i$ suits $|D_i| = h$; since $(N+1)N = T - 1$, $D_i$ is a $(T, h, 1)$-Difference Set. In [4], this is called a *Singer* Difference Set and it can be constructed only when $N$ is a prime power. Thus when $N \geq 3$ and $N$ is a prime power, $T \geq N^2 + N + 1$;
*Case* 3: If $T \geq N^2 + N + 2$ and $N$ is not a prime power, suppose an RDS $D_i$ suits $|D_i| = h$. It is clear that there are at most $h(h-1)$ ordered pairs $(a_j, a_k)$ and the difference values $a_j - a_k \equiv d \pmod{n}$ cannot cover $\{1, 2, \ldots, T-1\}$ since $h(h-1) = N(N+1) < N^2 + N + 1 \leq T - 1$, which implies $D_i$ is not an RDS under $Z_n$, and so $h \geq N + 2$. From

$$D_0 \bigcup D_1 \bigcup \cdots \bigcup D_{N-1} \subseteq Z_T \tag{8.34}$$

we can conclude

$$T \geq \sum_{i=0}^{N-1} |D_i| \geq Nh \geq N(N+2) \tag{8.35}$$

Therefore, the theorem holds.

The lower bound is not always tight. Finding the minimum *good* CCHS length is (almost) equivalent to finding the maximum DRDS. As discussed above, it is hard to find the maximum DRDS, and thus it is also hard to find the tight lower bound for *good* CCHS. From Table 8.2, the lower bound of Theorem 8.5 is tight when $N = 1, 2, 5, 6$. However, when $N = 3, 4$, the lower bound for $T$ is $13, 21$ respectively from the theorem, but the maximum DRDS $|S_n| = 2, 3$ under $Z_n$, implying the lower bound is not always tight.

**Corollary 8.1** *Any GS based rendezvous algorithm cannot guarantee rendezvous in less than $T$ time slots, where $T$ is the expression in Theorem 8.5.*

**Corollary 8.2** *The DRDS based algorithm (Algorithm 8.2) can achieve constant approximation as compared with the lower bound of any GS based blind rendezvous algorithms. Thus, it is a nearly optimal asynchronous rendezvous algorithm.*

*Remark 8.3* We have not found a general method to construct a DRDS $S$ under any $Z_n$ such that $|S|$ is comparable to the bound in Lemma 8.5. However, if there exists such DRDS construction for arbitrary $Z_n$, we can transform it to a good rendezvous algorithm as shown in Lemma 8.2.

## 8.7 Chapter Summary

In this chapter, we study the blind rendezvous problem for two users by designing a good *global sequence (GS)* that is independent of the user's set of available ports. The intuitive idea is to hop through the external ports by repeating the GS for the users, and rendezvous can be guaranteed on some common available port if the GS satisfies some good properties.

In order to design a good GS for the users, we introduce an efficient mathematical tool called Disjoint Relaxed Difference Set (DRDS) which is shown to be equivalent to a good GS. Therefore, every algorithm that constructs a DRDS under the set $Z_n$ in an efficient way can be adopted to construct a GS (of length $n$). In the chapter, we present a special construction of the DRDS under $Z_n$ when $n = 3P^2$ where $P$ is a prime number, and a good GS of length $3P^2 = O(N^2)$ can be reconstructed correspondingly, where $N$ is the number of all ports and $P \geq N$ is a prime number. Therefore, blind rendezvous between two users is guaranteed in $O(N^2)$ time slots, which is the state-of-the-art result for GS based method.

Since the users in the network can start the rendezvous process freely and they may have different sets of available ports, the DRDS based rendezvous algorithm works under all the situations:

(1) *Port-Symmetric* and *Port-Asymmetric*: two symmetric users have the same set of available ports, while the sets for asymmetric users could be different;
(2) *Synchronous* and *Asynchronous*: two synchronous users start at the same time while asynchronous users are free to start the rendezvous process.

The DRDS based rendezvous algorithm guarantees fast rendezvous for two symmetric users by adding a listening stage, where the user accesses the available port with the smallest label for a sufficient long time. Moreover, the algorithm works for two asynchronous users because the GS satisfies the elegant property (Property 8.2) and it guarantees rendezvous in a very short time when the users are synchronous. The results for the four combinations are listed in Table 8.3.

Although the DRDS based rendezvous algorithm has good performance, we are eager to explore a general method to construct DRDS under $Z_n$ where $n$ is an arbitrary integer and to design a GS based algorithm which generates a good GS of length shorter than $3P^2$.

**Table 8.3** *MTTR* values for the DRDS based rendezvous algorithm

| DRDS | Symmetric | Asymmetric |
|---|---|---|
| Synchronous | 1 | $2P = O(N)$ |
| Asynchronous | $3P = O(N)$ | $P(3P-1) = O(N^2)$ |

Remarks: 1) Improved DRDS algorithm guarantees rendezvous in $O(1)$ time slots for two symmetric users

## References


1. Bian, K., Park, J.-M. & Chen, R. (2009). A quorum-based framework for establishing control channels in dynamic spectrum access networks. In *Mobicom*.
2. Bian, K., Park, J.-M., & Chen, R. (2011). Control channel establishment in cognitive radio networks using channel hopping. *IEEE Journal on Selected Areas in Communications*, *29*(4), 689–703.
3. Chen, S., Russell, A., Samanta, A., & Sundaram, R. (2014). Deterministic blind rendezvous in cognitive radio networks. In *ICDCS*.
4. Colbourn, C. J., & Dintiz, J. H. (2006). *Handbook of Combinatorial Designs*. Boca Raton: CRC Press.
5. DaSilva, L., Guerreiro, I. (2008). Sequence-based rendezvous for dynamic spectrum access. In *DySPAN*.
6. Gu, Z., Hua, Q.-S., Wang, Y., Lau, F. C. M. (2013). Nearly optimal asynchronous blind rendezvous algorithm for cognitive radio networks. In *SECON*.
7. Hazan, E., Safra, S., & Schwartz, O. (2006). On the complexity of approximating $k$-set packing. *Computational Complexity*, *15*(1), 20–39.
8. Jiang, J. R., Tseng, Y. C., & Lai, T. (2005). Quorum-based asynchronous power-saving protocols for IEEE 802.11 ad hoc network. *ACM Journal on Mobile Networks and Applications*, *10*(1–2), 169–181.
9. Karp, R. M. (1972). Reducibility among combinatorial problems. *Complexity of Computer Computations*, 85–103.
10. Lin, Z., Liu, H., Chu, X., & Leung, Y.-W. (2013). Enhanced jump-stay rendezvous algorithm for cognitive radio networks. *IEEE Communications Letters*, *17*(9), 1742–1745.
11. Liu, H., Lin, Z., Chu, X., & Leung, Y.-W. (2012). Jump-stay rendezvous algorithm for cognitive radio networks. *IEEE Transactions on Parallel and Distributed Systems*, *23*(10), 1867–1881.
12. Luk, W. S., & Wong, T. T. (1997). Two new quorum based algorithms for distributed mutual exclusion. In *ICDCS*.
13. Shin, J., Yang, D., & Kim, C. (2010). A channel rendezvous scheme for cognitive radio networks. *IEEE Communications Letters*, *14*(10), 954–956.
14. Theis, N. C., Thomas, R. W., & DaSilva, L. A. (2011). Rendezvous for cognitive radios. *IEEE Transactions on Mobile Computing*, *10*(2), 216–227.
15. Yang, D., Shin, J., & Kim, C. (2010). Deterministic rendezvous scheme in multichannel access networks. *Electronics Letters*, *46*(20), 1402–1404.