ETCH: Efficient Channel Hopping for communication rendezvous in dynamic spectrum access networks — Source link

Yifan Zhang, Qun Li, Gexin Yu, Baosheng Wang

Institutions: College of William & Mary, National University of Defense Technology

Published on: 10 Apr 2011 - International Conference on Computer Communications

Topics: Rendezvous, Throughput, Radio resource management, Cognitive radio and Channel (programming)

WM-CS-2012-03

Toward Efficient Channel Hopping for Communication Rendezvous in Dynamic Spectrum Access Networks

Yifan Zhang, Gexin Yu, Qun Li, Haodong Wang,
Xiaojun Zhu and Baosheng Wang

June 18, 2012

Toward Efficient Channel Hopping for Communication Rendezvous in Dynamic Spectrum Access Networks

Yifan Zhang*, Gexin Yu*, Qun Li*, Haodong Wang†, Xiaojun Zhu‡§ and Baosheng Wang$^{\sharp\S}$
*The College of William and Mary, USA
†Cleveland State University, USA
‡Nanjing University, China
$^\sharp$National University of Defense Technology, China

***Abstract*—We present ETCH, efficient channel hopping based MAC-layer protocols for communication rendezvous in Dynamic Spectrum Access (DSA) networks. Compared to the existing solutions, ETCH fully utilizes spectrum diversity in communication rendezvous by allowing all the rendezvous channels to be utilized at the same time. We propose two protocols, SYNC-ETCH, which is a synchronous protocol assuming DSA nodes can synchronize their channel hopping processes, and ASYNC-ETCH, which is an asynchronous protocol not relying on global clock synchronization. Our theoretical analysis and `ns-2` based evaluation show that ETCH achieves better performances of time-to-rendezvous and throughput than the existing work.**

I. Introduction

Dynamic spectrum access (DSA) is a promising technique that solves the spectrum scarcity problem and increases network capacity. In DSA networks, unlicensed users (i.e., secondary users) are granted the access to the licensed spectrum if it is not being used by the licensed users (i.e., primary users). As in normal multi-channel communication networks, *communication rendezvous* is the first step for a pair of DSA nodes (i.e., secondary users[1]) to establish the communications with each other. In particular, a pair of DSA network nodes wishing to communicate should first agree on certain control information, such as data communication channel and data rate, before they can start the data communication. The channel on which the nodes negotiate to reach the agreement is called the *control channel*. Communication rendezvous for a pair of nodes is to establish a control channel between them.

The common control channel approach, where a well-known channel is designated as control channel for all nodes, is the most straightforward way to establish a control channel between a pair of DSA nodes. However, it suffers from the channel congestion problem and is vulnerable to jamming attacks [1]. Moreover, this approach cannot be applied in DSA networks because the control channel itself may be occupied by the primary user and hence become unavailable to the secondary users. The channel hopping approach, by contrast, increases control channel capacity and is immune to jamming attacks by utilizing multiple control channels. In this approach, all idle network nodes hop on a set of sequences of *rendezvous channels* (i.e., channels that are assigned for the purpose of control information exchange). When two nodes wishing to communicate hop to the same channel, this channel will serve as a control channel between the pair of nodes. The time that it takes for a pair of nodes to establish the control channel is called "*time-to-rendezvous*" or *TTR* for short.

To establish a control channel in DSA networks through channel hopping (abbreviation CH), every pair of nodes should have chance to rendezvous with each other periodically. In particular, due to the unique property of DSA networks that the channel availability is dynamic, the control channel established between any pair of nodes should equally likely be any one of the rendezvous channels. Otherwise, a pair of nodes would not be able to communicate if a primary user occupies the channels on which they rendezvous, even though there may still exist some other available channels to exchange the control information. QCH [2] is a recent control channel establishment protocol specifically designed for DSA networks. It utilizes the overlap property of quorums in a quorum system to develop CH sequences such that any two CH sequences are able to rendezvous periodically. Meanwhile, to accommodate the dynamics of the channel availability in DSA networks, QCH guarantees that any two nodes can meet each other as long as there are rendezvous channels not being occupied by primary users. While QCH is more suitable for DSA networks scenario and has better performances than existing CH-based multi-channel communication protocols, the following two concerns motivated us to explore for a better scheme.

First, in the scenario where global clock synchronization is available for DSA nodes to synchronize their channel hopping processes, QCH is only able to use one rendezvous channel as control channel in each hopping slot. This approach neglects the spectrum diversity, which is the most salient advantage brought by the DSA technique, in control channel establishment, and thus will potentially lead to severe traffic collision in a high probability. We propose SYNC-ETCH, a synchronous ETCH protocol, which efficiently exploits the spectrum diversity in a way that *every* rendezvous channel can serve as a control channel in *each* hopping slot. In SYNC-ETCH, while achieving the same goal, two CH sequence construction algorithms are proposed: two-phase CH sequence construction [3] and single-phase sequence construction. These two algorithms are complementary in design. The single-phase algorithm can *guarantee* the satisfaction of the even

[§]This work was done while the author was visiting the College of William and Mary.

[1]In this paper, DSA nodes always refer to secondary users of the DSA network.

use of rendezvous channels requirement, which states that all the rendezvous channels should have the same probability to appear in each constructed CH sequence. This requirement is important for CH based communication rendezvous protocols, since if a CH sequence is heavily using a certain rendezvous channel, the nodes hopping on this sequence will lose contact with other nodes if the heavily relied channel is taken away by the primary user. The constraint of the single-phase algorithm is that it requires the total amount of rendezvous channels to be an odd number. The two-phase CH sequence construction algorithm can be applied to DSA networks with an arbitrary number of rendezvous channel, but it tries (cannot guarantee) to satisfy the even use of rendezvous channel requirement. As will be showed later, both of the SYNC-ETCH CH sequence construction algorithms achieve the optimal average TTR under the premise that all the rendezvous channels should be utilized as control channels in every hopping slot.

Second, in the scenario where the channel hopping processes of different DSA nodes are not synchronized, QCH only guarantees two of the rendezvous channels to be used as control channels. This arrangement also does not take advantage of spectrum diversity in DSA networks, and may lead to communication outage when the primary users appear on the two channels. We propose ASYNC-ETCH, an asynchronous ETCH protocol, which solves the problems by using all rendezvous channels as control channels.

The contributions of this paper are summarized as follows.

- We formulate the problem of designing channel hopping based communication rendezvous protocols in DSA networks by considering all relevant metrics and requirements. We provide an in-depth and systematical analysis about the principles of designing this type of protocols, which is valuable for future research in this field.
- We propose an optimal synchronous protocol for communication rendezvous in DSA networks. The optimality of this protocol lies in that its average time-to-rendezvous is shortest under the premise that all the rendezvous channels should be utilized in every hopping slot. This approach achieves good time-to-rendezvous while greatly increasing the capacity of the DSA network at the communication setup stage.
- We propose a novel asynchronous protocol that enables two DSA network nodes to rendezvous without the existence of global clock synchronization mechanisms. Our protocol achieves better time-to-rendezvous and traffic throughput than the existing schemes.

The rest of the paper is organized as follows. We summarize the related work in section II. In section III, we formulate the problem to address and summarize the requirements for designing channel hopping based communication rendezvous protocols in DSA networks. We introduce the SYNC-ETCH protocol and the ASYNC-ETCH protocol in section IV and section V respectively. We theoretically compare the ETCH protocols with existing solutions in section VI, evaluate the performance of the ETCH protocols using `ns-2` simulations in section VII, and finally conclude the paper in section VIII.

II. Related Work

Channel hopping based rendezvous protocols in normal multi-channel wireless networks. SSCH [4] is a well known synchronous communication rendezvous protocol for IEEE 802.11 network. In SSCH, each node hops on a sequence of channels determined by multiple (channel, seed) pairs. The arrangement of the hopping sequence ensures that any two nodes have chance to rendezvous with very high probability. In very small chance that two nodes will never meet, a parity slot with fixed channel is introduced to allow the two nodes to communicate. CHMA [5] is another synchronous CH based rendezvous protocol. It directs all nodes to hop on a common channel sequence such that any two nodes can communicate while utilizing all the channels. These protocols for normal multi-channel wireless networks do not take into account some important properties of DSA networks, e.g., dynamic availability of channels, and thus are not suitable to be applied in DSA networks. Moreover, these protocols do not consider exploiting spectrum diversity in each hopping slot. DSA networks usually have much more spectrum diversity than normal multi-channel wireless networks. Therefore, exploiting spectrum diversity in DSA networks will bring much more performance gain.

Spectrum sharing in DSA networks. DSA network research can be divided into the following areas [6]: spectrum sensing ([7]–[12]), spectrum management ([12], [13]), spectrum mobility and spectrum sharing. Our work belongs to the area of spectrum sharing. In this area, techniques can be categorized into two classes based on network architecture. Techniques in the first class assume there is a centralized entity that is responsible for the spectrum allocation for all the secondary users in the network. DSAP [14] is a typical solution that belongs to this category. The second class of spectrum sharing techniques perform the sharing in a distributed manner. These techniques can be further divided into two groups based on the assumption about the existence of a common control channel. Techniques in the first group (e.g., DOSS [15]) use common control channels that are available to all secondary users for spectrum sharing information exchange. The second group of techniques, which do not rely on common control channel, allow DSA nodes rendezvous with each other and exchange spectrum sharing information in a dynamic manner. Among these techniques, some are based on channel hopping (detailed next) and some are not. HD-MAC [1] is a representative distributed technique that ensure rendezvous in DSA networks not based on channel hopping. In this scheme, secondary users self-organize into groups based on similarity of available channels. In each of the groups, a group control channel, elected by group members, is used to carry control information of the group nodes. A weakness of HD-MAC is that it relies on all-channel broadcast to spread spectrum availability information and control channel votes. Both sender and receiver of a broadcast message need to rotate on all their available channels to send or receive the message, which will take a long time in establishing the group control channel especially when the number of channels is high.

Channel hopping based rendezvous protocols in DSA networks. Our work, QCH [2], SeqR [16] and Jump-stay CH

[17] are representative CH based rendezvous protocols in DSA networks. QCH [2] deal with communication rendezvous in both the synchronous scenario and the asynchronous scenario, while SeqR [16] and Jump-stay CH [17] only deal with the asynchronous scenario. Different from the previous work, we focus on exploiting the spectrum diversity, which is the most salient advantage of DSA networks, in designing communication rendezvous protocols (for both scenarios).

III. Problem Formulation

A. Problem setting

In a DSA network, there are N (orthogonal) licensed channels labeled as $C_0, C_1, ..., C_{N-1}$ that can be used for control information exchange. In other words, there are N *rendezvous channels* in the DSA network. Any pair of nodes wishing to communicate with each other should first establish a control channel between them before data communications. We assume that there is no centralized entity that globally controls the allocation of communication channels, so the control channel establishment between a pair of nodes is executed in a distributed manner.

In a CH-based solution, idle nodes[2] **periodically** hop on (i.e., switch their working channel according to) a *CH sequence*, which is a sequence of rendezvous channels. The time during which a node stays on a channel is defined as a *hopping slot*, which is notated as a (slot-index, channel) pair. Thus, a CH sequence S is notated as

$$S = \{(0, S[0]), (1, S[1]), ..., (i, S[i]), ..., (p-1, S[p-1])\},$$

where $i \in [0, p-1]$ is the index of a hopping slot, and $S[i] \in \{C_0, \cdots, C_{N-1}\}$ is the rendezvous channel assigned to the i-th slot of the sequence S. The time it takes for a node to hop through the entire CH sequence is called a *hopping period*. When two nodes hop to the same channel, they can hear from each other and that channel is established as their control channel. If more than two nodes hop to the same rendezvous channel at the same time, they use existing collision avoidance mechanisms (e.g. RTS/CTS) or retransmission to establish pairwise control channels between them.

A CH-based solution should take account of the following requirements in its design.

- **Overlap requirement**. This requirement requires that any two CH sequences must overlap at a certain slot to ensure the rendezvous between the two nodes. Formally, given two CH sequences S_0 and S_1, they *overlap* if there exists a slot $(i, S_0[i]) \in S_0$ and a slot $(i, S_1[i]) \in S_1$ such that $S_0[i] = S_1[i]$. This slot is called an *overlapping slot* between S_0 and S_1, and the rendezvous channel $S_0[i]$ ($S_0[i] \in \{C_0, \cdots, C_{N-1}\}$) is called an *overlapping channel* between S_0 and S_1. If a rendezvous channel serves as an overlapping channel between a pair of CH sequences in the i-th slot, we say that *the rendezvous channel is utilized (as a control channel) in the i-th slot.*
- **Full utilization of rendezvous channels**. This requirement requires that any pair of nodes should be able to utilize every rendezvous channel as their control channel. This is to ensure the nodes have an opportunity to communicate with each other even if some of (but not all) the rendezvous channels are occupied by primary users.
- **Even use of rendezvous channels**. This requirement requires that all the rendezvous channels should have the same probability to appear in each CH sequence. If a CH sequence heavily relies on a certain channel (i.e., the channel is assigned to most of the slots of the CH sequence), nodes that hop on this CH sequence will lose contact with most of other nodes when the heavily relied channel is occupied by the primary user.

[2]Here idle nodes refer to nodes waiting to initiate a communication with other nodes and nodes waiting others to connect to them.

B. Metrics

We use the following three metrics in our numerical analysis for the proposed ETCH scheme.

- **Average rendezvous channel load.** This metric measures the average fraction of nodes that meet in the same rendezvous channel among all the nodes. Given a DSA network with M nodes and an average rendezvous channel load α $(0 < \alpha \leq 1)$, there are on average $M\alpha$ nodes rendezvous in the same channel. A light rendezvous channel load alleviates traffic collisions and increases the communication throughput.
- **Average time-to-rendezvous.** This is the average number of hopping slots that two nodes need to wait before they can rendezvous. A smaller average time-to-rendezvous (TTR) allows nodes to rendezvous and establish a communication link more quickly.
- **Rendezvous channel utilization ratio.** This is the ratio of the number of rendezvous channels that can be utilized as control channels in a hopping slot to the total number of rendezvous channels. It measures, in a given hopping slot, the extent that a communication rendezvous protocol utilizes the spectrum diversity in establishing control channels. A high rendezvous channel utilization ratio is helpful to reduce collision and improve the network capacity at the communication setup stage. This metric does not apply to the asynchronous case in which the hopping slot boundaries are not necessarily aligned.

We also use two other metrics, traffic throughput and actual time-to-rendezvous, to evaluate the practical performance of a communication rendezvous protocol. We will show that ETCH outperforms the existing solutions through mathematical analysis and simulations in section VI and section VII respectively.

C. Assumptions

We have the following assumptions regarding DSA networks and the node hardware.

- All the rendezvous channels are known to all the nodes. Information about rendezvous channels of a DSA network can be announced by regulation authorities such that all secondary users wishing to join the network will have this information.
- Each node is equipped with a single transceiver, which means a node cannot communicate in multiple channels at the same time. This assumption is in accordance with the ability of most commodity wireless devices.

- The channel switching overhead is negligible. This assumption is valid because most wireless hardware manufacturers claim that the channel switching delay is of the order of 80-90 μs [18]. This delay is negligible compared to the length of a slot in a hopping sequence, which is in the magnitude of 10ms.

IV. SYNC-ETCH

SYNC-ETCH assumes that there exists a synchronization mechanism to achieve global clock synchronization among DSA nodes, so that two nodes wishing to communicate with each other can start channel hopping at the same time.

A newly joined node execute the SYNC-ETCH protocol in following two steps.

In the first step, the node constructs a set of CH sequences by using either the *two-phase CH sequence construction* algorithm (Section IV-A) or the *single-phase CH sequence construction* algorithm (Section IV-B). The two-phase algorithm can be applied to scenarios with arbitrary numbers of rendezvous channels. It satisfies the overlap requirement in the first phase, and *tries* to fulfill the requirement of even use of rendezvous channels in the second phase. The single-phase algorithm *guarantees* the satisfaction of both requirements in an integral design. Both of the algorithms achieve the optimal average TTR under the premise that all the rendezvous channels should be utilized as control channels in every hopping slot.

The key design goal of both CH sequence construction algorithms is to fully utilize *all* the rendezvous channels in every hopping slot.

Theorem 1: In a DSA network with N rendezvous channels, for any CH based synchronous communication rendezvous protocols where *all* the rendezvous channels are utilized in each hopping slot, the minimum number of hopping slots of each CH sequence is $2N-1$, and the average TTR is $\frac{2N-1}{2}$.

Proof: To let all the N rendezvous channels be fully utilized in each CH time slot, we must arrange at least $2N$ CH sequences in a way that N pairs of CH sequences rendezvous at N different channels. We also must arrange at least $2N-1$ hopping slots for each of the $2N$ CH sequences to allow each sequence to rendezvous with the rest $2N-1$ CH sequences (for the overlap requirement). Considering that the rendezvous time of two randomly selected CH sequences (from the $2N$ sequences) is uniformly distributed between slot one and slot $2N-1$, the average TTR is $\frac{2N-1}{2}$. ■

Theorem 1 reveals that, to fully utilize all the N rendezvous channels in each hopping slot, there are at least $2N-1$ hopping slots in each CH sequence. As we will show later, both CH sequence construction algorithms in SYNC-ETCH can achieve such optimal length of CH sequences.

In the second step, the node starts the CH sequence execution process (Section IV-C) in a way that the full utilization of rendezvous channels requirement is satisfied.

We introduce the CH sequence construction algorithms and the CH sequence execution algorithm in the rest of this section.

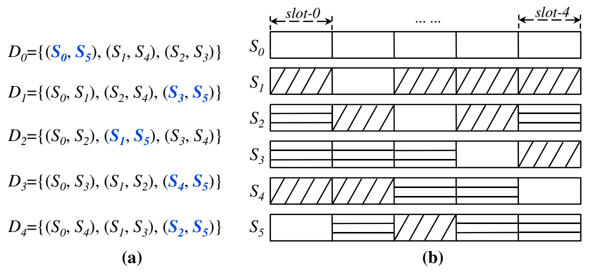
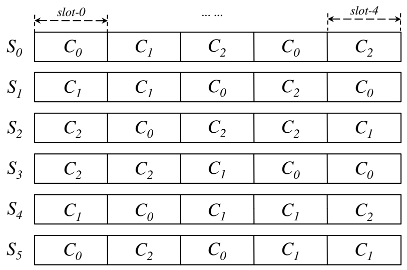

D_0={(S_0, S_5), (S_1, S_4), (S_2, S_3)}
D_1={(S_0, S_1), (S_2, S_4), (S_3, S_5)}
D_2={(S_0, S_2), (S_1, S_5), (S_3, S_4)}
D_3={(S_0, S_3), (S_1, S_2), (S_4, S_5)}
D_4={(S_0, S_4), (S_1, S_3), (S_2, S_5)}
slot-0
... ...
slot-4
S_0
S_1
S_2
S_3
S_4
S_5
(a)
(b)

Fig. 1. Phase 1 of the two-phase CH sequence construction - *rendezvous scheduling*. This figure shows the 5 rendezvous schedules (D_0 to D_4) of a DSA network with 3 rendezvous channels (i.e., $N = 3$). In this network, 6 CH sequences (S_0 to S_5) are constructed. Each CH sequence has 5 hopping slots. Rendezvous schedule $D_i(0 \leq i \leq 4)$ specifies how nodes following two different CH sequences rendezvous in the slot-i. For instance, in slot-0, the nodes hopping on CH sequence S_0 meet the nodes on S_5 in one of the 3 rendezvous channels, the nodes on S_1 meet the nodes on S_4 in a different rendezvous channel, and the nodes on S_2 meet the nodes on S_3 in the remaining channel.


slot-0
... ...
slot-4

S_0	C_0	C_1	C_2	C_0	C_2
S_1	C_1	C_1	C_0	C_2	C_0
S_2	C_2	C_0	C_2	C_2	C_1
S_3	C_2	C_2	C_1	C_0	C_0
S_4	C_1	C_0	C_1	C_1	C_2
S_5	C_0	C_2	C_0	C_1	C_1

Fig. 2. Phase 2 of the two-phase CH sequence construction - *rendezvous channel assignment*. This figure shows the 6 final CH sequences of a DSA network with 3 rendezvous channels C_0, C_1 and C_2 (i.e., $N = 3$). A greedy algorithm is used to assign the 3 rendezvous channels to each slot (slot-0 to slot-5) of all the 6 CH sequences (S_0 to S_5) based on the rendezvous schedules output by the rendezvous scheduling phase. For instance, nodes following the CH sequence S_0 hop on a sequence of channels $C_0 \rightarrow C_1 \rightarrow C_2 \rightarrow C_0 \rightarrow C_1$ periodically.

A. Two-phase CH sequence construction

1) An overview and an example: The two-phase CH sequence construction algorithm constructs a set of CH sequences in two phases.

The first phase is called the *rendezvous scheduling* phase. In this phase, the algorithm creates a set of rendezvous schedules, each of which instruct how nodes with different CH sequences meet with each other in a hopping slot. Given a DSA network with N rendezvous channels, to fully utilize spectrum diversity, an ideal rendezvous schedule allows N pairs of nodes to rendezvous at N different channels in a hopping slot, which is equivalent to arrange for $2N$ CH sequences (each of which is used by one participating node) to overlap at different N channels in a slot. Meanwhile, the rendezvous schedules should ensure the satisfaction of the overlap requirement, i.e., any pair of nodes hopping on different CH sequences can meet at least once within a hopping period.

The second phase is called the *rendezvous channel assignment* phase. In this phase, the algorithm fills the rendezvous channels in the $2N$ CH sequences based on the rendezvous schedules generated in the previous phase. This phase tries to satisfy the design requirement of even use of rendezvous channels by using a greedy algorithm. At the end of the rendezvous channel assignment phase, $2N$ CH sequences are constructed.

Fig. 1 and Fig. 2 illustrate an example of the two-phase CH sequence construction in a DSA network with three ($N = 3$) rendezvous channels. Fig. 1(a) shows the five rendezvous schedules $D_0, D_1, ..., D_4$ generated in the rendezvous scheduling phase. Each rendezvous schedule corresponds one of the five $(2N-1)$ hopping slots of the six $(2N)$ the CH sequences $S_0, S_1, ..., S_5$. As we can see, in each of the five hopping slots, six nodes (selecting different CH sequences) are supported to rendezvous in three different channels. For example, in the first slot (i.e., slot-0), where the rendezvous schedule D_0 is used to arrange rendezvous, the node selecting CH sequence S_0 is arranged to rendezvous with the node selecting CH sequence S_5 on one rendezvous channel, while the node selecting S_1 meets with the node selecting S_4 on a different rendezvous channel, and the node selecting S_2 meets with the node selecting S_3 on the remaining rendezvous channel. Fig. 1(b) shows the overall effect of how the six node selecting different CH sequences rendezvous in different hopping slot. In each hopping slot in Fig. 1(b), a pair of nodes whose CH sequences have the same type of shade will meet on the same rendezvous channel. Please note that the detailed arrangement about rendezvous channels on which pairs of nodes rendezvous has not yet been determined in this phase, and is left to the next phase. As will be presented shortly, our algorithm schedules the $2N$ CH sequences to meet in the N different rendezvous channels in a hopping slot as follows. It selects $2N-2$ out of the first $2N-1$ CH sequences (i.e., $S_0, \cdots, S_{2N-2}$) to form $N-1$ CH sequences pairs, where each sequence is scheduled to meet the other sequence from the same pair in the slot-sl, such that the index sum of each pair of CH sequences is congruent to sl modulo $2N-1$. The remaining CH sequence (within $S_0, \cdots, S_{2N-2}$) and S_{2N-1} form the last pair of CH sequences (shown in blue color in Fig 1(a)) that are scheduled to meet in the slot-sl.

Fig. 2 shows an example of rendezvous channel assignment once the scheduling is determined. From the example we can see that all the rendezvous channels are utilized for communication in each of the hopping slots, and that each rendezvous channel appears in each of the CH sequences with roughly the same probability.

2) Phase 1: rendezvous scheduling: We now formalize the problem of rendezvous scheduling, the first phase of the two-phase CH sequence construction process, as follows. Given a set of $2N$ CH sequences $U = \{S_0, S_1 \cdots, S_{2N-1}\}$, $D_{sl} = \{d_0, d_1, \cdots, d_{N-1}\}$ is called a *rendezvous schedule* for the hopping slot indexed in sl if $\bigcup D_{sl} = d_0 \cup d_1 \cup \cdots \cup d_{N-1} = U$, where $d_i = \{S_a, S_b\}$ $(0 \leq i \leq N-1)$ is a pair of CH sequences that are scheduled to rendezvous in the slot-sl.

According to theorem 1, the optimal rendezvous scheduling algorithm must construct $2N-1$ different rendezvous schedules, each of which corresponds to a hopping slot, such that each CH sequence is able to rendezvous with all the other $2N-1$ CH sequences in $2N-1$ hopping slots. SYNC-ETCH uses Algorithm 1 to construct the schedules.

In Algorithm 1, rendezvous schedule D_{sl} $(0 \leq sl \leq 2N-2)$, which is the rendezvous schedule for the slot-sl, is constructed as follows. Within the CH sequences set $V = \{S_0, \cdots, S_{2N-2}\}$, S_a and S_b are scheduled to rendezvous in the slot-sl (i.e., $\{S_a, S_b\} \in D_{sl}$) if $a + b \equiv sl(\bmod(2N-1))$

Algorithm 1: Rendezvous Scheduling

```
Data: $U = \{S_0, \cdots, S_{2N-1}\}$: a set of $2N$ empty CH
      sequences, each of which has $2N-1$ slots;
Result: $\mathcal{D} = \{D_0, D_1, \cdots, D_{2N-2}\}$: $2N-1$ different
        rendezvous schedules of $U$.
1  Initialize $D_0, D_1, \cdots, D_{2N-2}$ to be empty;
2  for $sl \leftarrow 0$ to $2N-2$ do
3      $V \leftarrow U \backslash \{S_{2N-1}\}$;
4      for $i \leftarrow 0$ to $N-1$ do
5          $a \leftarrow$ the smallest subscript in $V$;
6          if $a \leq sl$ then
7              $b \leftarrow sl - a$;
8          else
9              $b \leftarrow 2N - 1 + sl - a$;
10         if $a == b$ then
11             $b \leftarrow 2N - 1$;
12         $d_i \leftarrow \{S_a, S_b\}$;
13         $D_{sl} \leftarrow D_{sl} \cup \{d_i\}$;
14         $V \leftarrow V \backslash \{S_a, S_b\}$;
15 return $D_0, D_1, \cdots, D_{2N-2}$;
```

and $a \neq b$. For the CH sequence $S_a \in V$ that satisfies $2a \equiv sl(\bmod(2N-1))$, it is scheduled to rendezvous with the CH sequence S_{2N-1} in the slot-sl (i.e., $\{S_a, S_{2N-1}\} \in D_{sl}$).

We now prove the correctness of Algorithm 1 as follows.

Theorem 2: Algorithm 1 constructs $2N-1$ rendezvous schedules of U, and all these $2N-1$ rendezvous schedules are different.

Proof: In order to prove Algorithm 1 constructs $2N-1$ rendezvous schedules, we need to prove given an integer sl $(0 \leq sl \leq 2N-2)$, D_{sl} is a rendezvous schedule of U. To prove this, we need to prove
`(1)` there is only a number $x \in [0, 2N-2]$ such that $2x \equiv sl \pmod{(2N-1)}$, and
`(2)` $\forall\, a, b, c, d \in [0, 2N-2]$ that satisfy $a + b \equiv sl \pmod{(2N-1)}$ and $c + d \equiv sl \pmod{(2N-1)}$, if $a \neq c$ then $b \neq d$.
By proving `(1)` we can guarantee that the CH sequence S_{2N-1} only exists in only a CH sequence pair d_i $(0 \leq i \leq N-1)$ within rendezvous schedule D_{sl}. From `(1)`, `(2)` and the strategy that we always choose the first CH sequence of d_i $(0 \leq i \leq N-1)$ from a set of CH sequences that have never been chosen (i.e. set V in Algorithm 1)(line 5), we can ensure that $\bigcup D_{sl} = d_0 \cup d_1 \cup \cdots \cup d_{N-1} = U$ (i.e. D_{sl} is a rendezvous schedule of U).

We prove both `(1)` and `(2)` by contradiction. For `(1)`, suppose there are two different number m and n that satisfy $0 \leq m < n \leq 2N-2$, $2m \equiv sl \pmod{(2N-1)}$ and $2n \equiv sl \pmod{(2N-1)}$, then we can have $2m = sl$ and $2n = 2N - 1 + sl$. A contradiction is found that sl is an even number because $2m = sl$, and sl is an odd number because $2n = 2N - 1 + sl$. For `(2)`, without loss of generality, we suppose $a < c$. If $b = d$, then we have $a + b = sl$ and $c + d = 2N - 1 + sl$. By subtracting these two equations we get $c - a = 2N - 1$ which is impossible because $0 \leq a < c \leq 2N - 2$.

Algorithm 2: Rendezvous Channel Assignment

```
Data: $\mathcal{C} = \{C_0, C_1, \cdots, C_{N-1}\}$: $N$ rend. channels;
      $\mathcal{D} = \{D_0, D_1, \cdots, D_{2N-2}\}$: $2N-1$ rendezvous
      schedules returned by Algorithm 1.
Result: $S_0, S_1, \cdots, S_{2N-1}$: $2N$ final CH sequences.
1  for $i \leftarrow 0$ to $2N-1$ do
2      $seqOC[i] \leftarrow \{C_0, C_1, \cdots, C_{N-1}\}$; /*Initializing the
       outstanding channels of the CH sequence $S_i$.*/
3  for $sl \leftarrow 0$ to $2N-2$ do
4      $slotOC \leftarrow \{C_0, C_1, \cdots, C_{N-1}\}$; /*Initializing the
       outstanding channels of the slot-sl.*/
5      for $n \leftarrow 0$ to $N-1$ do
6          Mark CH sequence pair $d_n \in D_{sl}$ as unassigned;
7      while $slotOC \neq \phi$ do
8          Pick $d_n = \{S_i, S_j\}$ from the unassigned CH
           sequence pairs in $D_{sl}$ such that $seqOC[i] +$
           $seqOC[j]$ is the greatest (if multiple choices exist,
           pick the pair that contains the CH sequence with the
           smallest index);
9          $k \leftarrow seqOC[i] \geq seqOC[j]\ ?\ i : j$;
10         if $slotOC \cap seqOC[k] \neq \phi$ then
11             $c \leftarrow$ the channel in $slotOC \cap seqOC[k]$ with the
               smallest index;
12             $seqOC[k] \leftarrow seqOC[k] \backslash \{c\}$;
13         else
14             $c \leftarrow$ the channel in $slotOC$ that appears the fewest
               times in $S_k$ (if multiple choices exist, pick the one
               with the smallest index);
15         $S_i \leftarrow S_i \cup (sl, c)$;
16         $S_j \leftarrow S_j \cup (sl, c)$;
17         Mark $d_n = \{S_i, S_j\}$ as assigned;
18         $slotOC \leftarrow slotOC \backslash \{c\}$;
19 return $S_0, S_1, \cdots, S_{2N-1}$;
```

In order to prove $\forall p, q \in [0, 2N-2]$ $(p \neq q)$, rendezvous schedule D_p and schedule D_q are different, we need to prove $\forall d_i \in D_p (0 \leq i \leq N-1)$ and $\forall d_j \in D_q (0 \leq j \leq N-1)$, $d_i \neq d_j$. We prove this by contradiction. Suppose there exist $d_i \in D_p$ and $d_j \in D_q$ such that $d_i = d_j$, which means d_i and d_j contain the same pair of CH sequences. Suppose these two sequences are S_u and S_v, where $0 \leq u, v \leq 2N-1$. Then we have $u + v \equiv p \pmod{(2N-1)}$ and $u + v \equiv q \pmod{(2N-1)}$, where $p, q \in [0, 2N-2]$ and $p \neq q$, which is impossible. ■

3) Phase 2: rendezvous channel assignment: In the second phase of the two-phase CH sequence construction process, we assign rendezvous channels to each of the $2N$ CH sequences according to the rendezvous schedules generated in the previous phase. The goal of the rendezvous channel assignment phase is two-fold. *First*, to fully exploit the frequency diversity of a DSA network in establishing control channels, *all* the rendezvous channels should be utilized in each hopping slot. *Second*, the assignment tries to satisfy the even use of rendezvous channels requirement presented in Section III-A by an arrangement that allows each rendezvous channel to have a roughly equal probability to appear in each CH sequence.

We employ a greedy algorithm (shown in Algorithm 2) to achieve the goals of rendezvous channel assignment. In Algorithm 2, rendezvous channels are assigned to CH sequences round by round (lines 3-18). In the sl-th $(0 \leq sl \leq 2N-2)$ round, the rendezvous channels are assigned to the slot-sl of all the CH sequences based on the slot's rendezvous schedule, D_{sl}, constructed in the previous phase. For each hopping slot, the algorithm needs to guarantee that every rendezvous channel is assigned to a pair of CH sequences (to achieve the first goal of rendezvous channel assignment). To keep track of the channel assignment for each slot, the variable $slotOC$ is used to record the outstanding rendezvous channels of the current slot, i.e., the rendezvous channels that have not been assigned to the slot. At the beginning of each round of channel assignment, $slotOC$ is reset to the whole set of rendezvous channels (line 4). The algorithm also tries to make all the rendezvous channels appear in each CH sequence with a roughly equal probability (to achieve the second goal of rendezvous channel assignment). To keep track of the channel assignment for each CH sequence, the variable $seqOC[i]$ $(0 \leq i \leq 2N-1)$ is used to record the outstanding rendezvous channels of the CH sequence S_i, i.e., the rendezvous channels that have not been assigned to the CH sequence S_i. The variables $seqOC[i]$ are initialized to the whole set of rendezvous channels (lines 1-2).

Before the sl-th round of rendezvous channel assignment (i.e., the round that assign channels to slot-sl of all the CH sequences), all the CH sequence pairs in D_{sl} are initially marked as "unassigned" (lines 5-6). In the sl-th round of rendezvous channel assignment, the algorithm checks all the unassigned CH sequence pairs in D_{sl}, and selects the pair $\{S_i, S_j\}$ such that the sum of outstanding channels of S_i and S_j is *greatest* compared to other unscheduled CH sequence pairs in D_{sl}. If there are multiple pairs that produce the same greatest outstanding channels sum, the pair that contains the smallest indexed CH sequence is selected (line 8). Then the algorithm chooses a rendezvous channel to assign to the slot-sl of both S_i and S_j (lines 9-16). This rendezvous channel is selected as follows. Within the CH sequences S_i and S_j, the one with more outstanding channels is notated as S_k (line 9). The rendezvous channel is first selected from the intersection of the slot-sl's outstanding channels (recorded in $slotOC$) and S_k's outstanding channels (recorded in $seqOC[k]$) (line 11). If the intersection is empty, the channel is selected from the slot-sl's outstanding channel that appears *fewest* times in S_k (line 14). Then this rendezvous channel is assigned to the slot-sl of both CH sequences of S_i and S_j (lines 15-16), and the CH sequence pair $\{S_i, S_j\}$ is marked as "assigned" (line 17). The selected rendezvous channel is removed from the current slot's outstanding channels set (line 18). It is also removed from S_k's outstanding channels if has not been assigned to S_k before the assignment (line 12).

Fig. 2 shows the result of rendezvous channel assignment in a DSA network with 3 rendezvous channels, C_0, C_1 and C_2. CH sequences S_0 to S_5 are the final CH sequences constructed by the two-phase CH sequence construction process. From the example we can see that all the rendezvous channels are utilized for communication in each of the hopping slots, and that each rendezvous channel appears in each of the CH sequences with roughly the same probability.

B. Single-phase CH sequence construction

1) An overview and an example: In a DSA network with N rendezvous channels, similar to the two-phase algorithm, the single-phase CH sequence construction algorithm constructs $2N$ CH sequences, each of which has $2N-1$ hopping slots, such that the following two requirements are satisfied. *First*, every CH sequence meets with all the other $2N-1$ CH sequences each at a time in a hopping slot. *Second*, all the rendezvous channels are utilized for CH sequence rendezvous in every hopping slot. The improvement of the single-phase algorithm over the two-phase algorithm is that it can guarantee to satisfy a *third* requirement that every rendezvous channel has the same probability to appear in each CH sequence (i.e., the even use of rendezvous channels requirement). For instance, in a DSA network with three rendezvous channels, the even use of rendezvous channels requirement expects there are two rendezvous channels appearing twice and the remaining channel appearing once in the five hopping slots of *every* CH sequence. However, in the six CH sequences constructed by the two-phase algorithm (shown in Figure 2), channel C_2 and C_1 appear three times in the CH sequence S_2 and S_4 respectively. By contrast, the single-phase algorithm can guarantee the even use of rendezvous channels requirement.

The single-phase CH sequence construction algorithm views the rendezvous among the CH sequences within a hopping period as a colored graph G. To satisfy the three requirements above, the colored graph G should have the following properties. *First*, there are $2N$ vertices in G:

$$|V(G)| = 2N, \tag{1}$$

where $V(G)$ denotes the vertex set of the graph G. Each vertex corresponds to one of the $2N$ CH sequences. *Second*, the edges in G have N different colors:

$$C(G) = \{c_0, ..., c_{N-1}\}, \tag{2}$$

where $C(G)$ denotes the color set on edges in the graph G. Each color corresponds to a rendezvous channel. If two CH sequences rendezvous in a certain channel, the corresponding pair of vertices in G are connected by an edge with the corresponding color. *Third*, since every CH sequence should rendezvous with all the other $2N-1$ sequences exactly once in a hopping period, the graph G should satisfy

$$\forall v \in V(G), d(v) = 2N-1, \tag{3}$$

where $d(v)$ denotes the degree of vertex v. In other words, the graph G should be a $2N$-vertex complete graph K_{2N}. *Fourth*, since all the rendezvous channels should appear in every CH sequence, the color degree of each vertex, which is the number of colors on the edges incident to the vertex, should be N:

$$\forall v \in V(G), \delta(v) = N, \tag{4}$$

where $\delta(v)$ is the number of colors on edges incident to the vertex v. *Fifth*, since each of the N rendezvous channel should have the same probability to appear in every CH sequence (i.e., the even use of rendezvous channels requirement), among the N different colors on the $2N-1$ edges incident to a vertex v, there should be one color to appear once and the remaining $N-1$ colors to appear twice, which is the best scenario satisfying the even use of rendezvous channels requirement:

$$\forall v \in V(G)\ (\exists c_i \in C(G)\ (\delta_{c_i}(v) = 1\ \ \&\& \\ \delta_{c_j}(v) = 2, \forall j \in [0, N-1], j \neq i)), \tag{5}$$

where $\delta_{c_i}(v)$ is the number of edges colored with c_i that are incident to the vertex v. Figure 4(a) shows an example of the graph G for a DSA network with 5 rendezvous channels (i.e., $N = 5$). The graph G in the example is a 5-colored 10-vertex complete graph K_{10}. In this graph, each of the 5 colors has an even probability to appear on the 9 edges that are incident to each vertex (i.e., one color appears once and each of the rest four colors appear twice), which is the best case of satisfying the even use of rendezvous channels requirement.

The graph G with the properties (1) to (5) tells how each of the $2N$ CH sequence meets with each other in the N rendezvous channels *within a hopping period*. The single-phase CH sequence construction algorithm needs to further specify how the CH sequences rendezvous with each other in each of the $2N-1$ hopping slots. To fully exploit the spectrum diversity, our algorithm ensures that all the rendezvous channels can be utilized as control channel in every hopping slot. This is achieved by decomposing the graph G into $2N-1$ *different perfect rainbow matchings*, each of which instructs how the $2N$ CH sequences rendezvous in a hopping slot. In graph theory, a *matching* in a graph G is a set of edges of G without common vertices, a *perfect matching* in G is a matching that covers all the vertices of the graph G [19], and a *rainbow matching* in G is a matching where edges have distinct colors [20]. Therefore, in our case, a *perfect rainbow matching* (notated as PRM) in a N-colored $2N$-vertex complete graph G is an edge set that contains N disjoint edges of G colored with the N distinct colors:

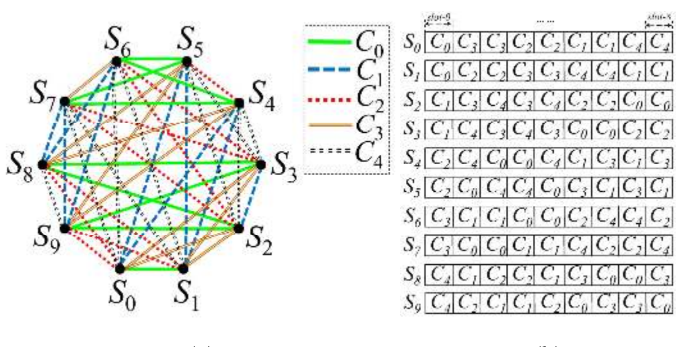
$$PRM = \{\hat{E} \mid V(\hat{E}) = V(G)\ \ \&\& \\ \forall e_i \in \hat{E}\ (\forall v \in V(e_i)\ (v \notin V(e_j), \forall e_j \in \hat{E}, j \neq i))\ \&\& \\ \forall e_i, e_j \in \hat{E},\ i \neq j\ (C(e_i) \neq C(e_j))\} \tag{6}$$

where $V(\hat{E})$ denotes the set of vertices of the edge set $\hat{E}$, $V(e_i)$ denotes the two vertices on the edge e_i, and $C(e_i)$ denotes the color on the edge e_i. Given two perfect rainbow matching PRM_i and PRM_j, they are different if and only if the there is no common edges in them:

$$PRM_i \text{ and } PRM_j \text{ are different } \texttt{iff} \\ \forall e_i \in E(PRM_i), e_j \in E(PRM_j)\ (e_i \neq e_j), \tag{7}$$

where $E(PRM_i)$ denotes the edge set in the perfect rainbow matching PRM_i. In our example, the 5-colored 10-vertex complete graph G shown in Figure 4(a) can be decomposed into 9 different *PRM*s shown in Figure 3(1), Figure 3(6a-1) to (6a-4), and Figure 3 (6b-1) to (6b-4) respectively. Each of these *PRM*s is the rendezvous schedule for one hopping slot within a hopping period. The final CH sequences (shown in Figure 4(b)) are constructed based on these 9 different *PRM*s.

In the following, we will show that, for a DSA network with N rendezvous channels, where N is an odd number greater than two, our single-phase CH sequence construction



	slot-0				……				slot-8
S_0	C_0	C_3	C_3	C_2	C_2	C_1	C_1	C_4	C_4
S_1	C_0	C_2	C_2	C_3	C_3	C_4	C_4	C_1	C_1
S_2	C_1	C_3	C_4	C_3	C_4	C_2	C_2	C_0	C_0
S_3	C_1	C_4	C_3	C_4	C_3	C_0	C_0	C_2	C_2
S_4	C_2	C_4	C_0	C_0	C_4	C_1	C_3	C_1	C_3
S_5	C_2	C_0	C_4	C_4	C_0	C_3	C_1	C_3	C_1
S_6	C_3	C_1	C_1	C_0	C_0	C_2	C_4	C_4	C_2
S_7	C_3	C_0	C_0	C_1	C_1	C_4	C_2	C_2	C_4
S_8	C_4	C_1	C_2	C_2	C_1	C_3	C_0	C_0	C_3
S_9	C_4	C_2	C_1	C_1	C_2	C_0	C_3	C_3	C_0

Fig. 4. For a DSA network with 5 rendezvous channels (i.e., $N = 5$), (a) is the N-colored $2N$-vertex complete graph G that shows how the $2N$ CH sequence rendezvous with each other within a hopping period, and (b) shows the final $2N$ CH sequences.

algorithm can form a graph G with the properties of (1) to (5), and decompose the graph G into $2N-1$ different perfect rainbow matchings (i.e., properties of (6) and (7)).

2) An intuitive description of the algorithm: For a DSA network with N rendezvous channels, where N is an odd number greater than two, the single-phase CH sequence construction algorithm constructs the $2N$ CH sequences as follows.

The initial state: We treat the $2N$ CH sequences to be constructed as $2N$ vertices in a graph G, and assign different colors to each of the N rendezvous channels. Initially, the graph G only contains the $2N$ vertices and no edges. Our goal is to add colored edges to G to make it an N-colored $2N$-vertex complete graph K_{2N} with the properties (1) to (5), and then decompose the K_{2N} into $2N-1$ different perfect rainbow matchings (i.e., properties (6) and (7)).

In the following, we notate the $2N$ CH sequences as $S_0 \cdots S_{2N-1}$, and the N rendezvous channels as $C_0, \cdots, C_{N-1}$. We will take a DSA networks with 5 rendezvous channels (i.e., $N = 5$) as an example throughout this subsection. Figure 3(0) shows the initial state of the graph G in the example.

Step #1 - forming the first perfect rainbow matching of K_{2N}: In the first step, we form the first PRM by connecting vertex S_i with vertex S_{i+1} by using an edge with the color of channel $C_{\frac{i}{2}}$, where i is an even number in the range of $[0, 2N-1]$. The PRM tells how the $2N$ CH sequences rendezvous with each other in the hopping slot-0. In our example, Figure 3(1) shows the state of the graph G after the step #1 is applied. It is a PRM of a $2N$-vertex complete graph K_{2N}, which specifies that in the hopping slot-0, CH sequences S_0 and S_1, S_2 and S_3, S_4 and S_5, S_6 and S_7, S_7 and S_8 rendezvous in channel C_0 to C_4 respectively.

Step #2 - shrinking the $2N$ **vertex graph** G **to** K_N: In the second step, we shrink the $2N$-vertex graph G to an N-vertex N-colored complete graph K_N as follows. First, we combine every connected vertex pair in the first PRM of K_{2N} (i.e., $\{S_i,\ S_{i+1}\}$, where $i = 2a, a \in [0, N-1]$) into a new vertex (notated as $S_{i,i+1}$), and connect the new vertices to each other to form a N-vertex complete graph K_N. Second, we give each vertex in K_N the color of the edge connecting the corresponding vertex pair in the first PRM of K_{2N}. Figure 3(2) shows the state of the graph G after the step #2 is applied.

Step #3 - decomposing K_N **into** $\frac{N-1}{2}$ **different 2-factors**: In graph theory, a 2-factor of a graph G is spanning subgraph of G, where the degree of each vertex in the subgraph is 2. Algorithm 3 decomposes K_N into $\frac{N-1}{2}$ 2-factors. In the algorithm, each edge of K_N and its vertices are put into $\frac{N-1}{2}$ graphs, $TF_1, \cdots, TF_{\frac{N-1}{2}}$, depending on the subscript difference between the two vertices: given an edge $e = (S_{i,i+1}, S_{j,j+1}) \in E(K_N)$, where $i < j$, the edge e and its vertices are put into the graph TF_d ($1 \le d \le \frac{N-1}{2}$) if either $\frac{j-i}{2}$ or $\frac{i+2N-j}{2}$ equals to d (line 5).

Algorithm 3: 2-factorization of complete graph K_N

```
Data: $K_N$ formed after step #2, the $N$ vertices of $K_N$
      are $\{S_{0,1}, \cdots, S_{2N-2,2N-1}\}$;
Result: $\frac{N-1}{2}$ 2-factors of $K_N$: $TF_1, \cdots, TF_{\frac{N-1}{2}}$.
1 for $d \leftarrow 1$ to $\frac{N-1}{2}$ do
2     $TF_d \leftarrow \phi$;
3     for each edge $(S_{i,i+1}, S_{j,j+1}) \in E(K_N)$ do
4         if $\frac{|i-j|\ \%\ N}{2} == d$ then
5             Add edge $(S_{i,i+1}, S_{j,j+1})$ and its vertices to $TF_d$;
6 return $TF_1, \cdots, TF_{\frac{N-1}{2}}$;
```

Theorem 3: Each graph TF_d ($1 \le d \le \frac{N-1}{2}$) decomposed from K_N using Algorithm 3 is a 2-factor of the complete graph K_N.

Proof: According to the algorithm, among all the edges that are incident to *each* $S_{2i,2i+1} \in V(K_N)$ ($0 \le i \le N-1$), only the edge ($S_{2i,2i+1}$,$S_{(2(i+d))\%2N,(2(i+d)+1)\%2N}$) and the edge ($S_{2i,2i+1}$,$S_{(2(i-d))\%2N,(2(i-d)+1)\%2N}$) and their associated vertices are added to the graph TF_d. Since $d \ne 0$ and $d \ne \frac{N}{2}$, we have $(2(i+d))\%2N \ne (2(i-d))\%2N$, which means the two edges added to TF_d are different. Therefore, $V(TF_d)$ equals to $V(K_N)$, and the degree of each vertex in TF_d is 2. Thus, TF_d is a 2-factor of the complete graph K_N.

Simple calculation can easily confirm that the 2-factor TF_d is either a Hamiltonian cycle of K_N if d and N is coprime, or $GCD(N,d)$ disjoint $\frac{N}{GCD(N,d)}$-cycles (i.e., cycles with $\frac{N}{GCD(N,d)}$ edges), where $GCD(N,d)$ is the greatest common divisor of N and d, otherwise. ■

It is obvious that the $\frac{N-1}{2}$ 2-factors are different, since each edge of K_N can only belong to one 2-factor.

In our example, Figure 3(3a) and Figure 3(3b) are the two 2-factors of the complete graph K_5 obtained after the step #2. They are both Hamiltonian cycles of K_5. In Appendix-A, we show a more general example where there exists a 2-factor of K_N being a spanning subgraph of disjoint cycles.

Step #4 - rainbow-coloring 2-factors of K_N: In the fourth step, we color each of K_N's 2-factors using all the N colors such that each edge will have a different color, which is a process called "rainbow-coloring".

Algorithm 4 shows the rainbow-coloring algorithm. In this algorithm, the color to put on a edge is the color of another vertex that is different from the two vertices of the edge. Specifically, the color to put on the edge $(S_{i,i+1}, S_{j,j+1})$ is the color of either the vertex $S_{\frac{i+j}{2},\frac{i+j}{2}+1}$ (if $\frac{i+j}{2}$ is an even number) or the vertex $S_{\frac{i+j+2N}{2}\%2N,\frac{i+j+2N}{2}\%2N+1}$ (if $\frac{i+j}{2}$ is an odd number). We have the following two theorems about the

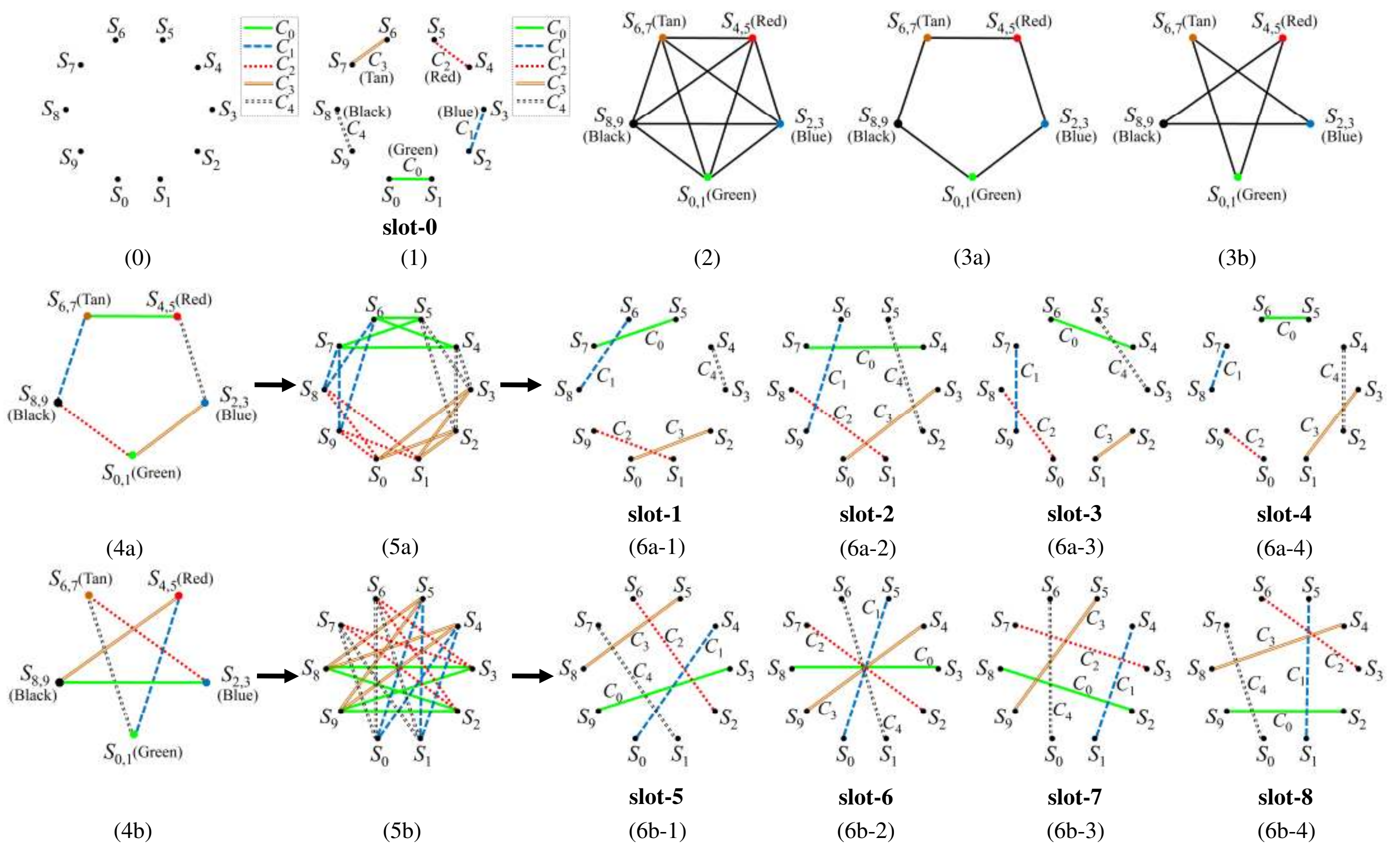


Fig. 3. The six steps of the intuitive description of the single-phase CH sequence construction algorithm for a DSA network with 5 rendezvous channels (i.e., $N = 5$). (0) is the initial state of the graph G; (1) shows how the the first PRM is formed in the step #1; (2) shows how the graph G is shrank to K_5 in the step #2 based on the first PRM; (3a) and (3b) are the two 2-factors of K_5 obtained in the step #3; (4a) and (4b) show the rainbow-coloring of the two 2-factors of K_5 in the step #4; (5a) and (5b) show how the two rainbow-colored 2-factors of K_5 are expanded back to the two 4-factors in K_{10} in the step #5; (6a-1) to (6a-4) and (6b-1) to (6b-4) are respectively the final 4 PRMs decomposed from the two rainbow-colored 4-factors of K_{10}.

Algorithm 4: Rainbow-coloring a 2-factor of K_N

```
Data: A 2-factor $TF$ obtained in step #3;
Result: Rainbow-coloring the 2-factor such that every
        edge's color is different from the colors of its
        two vertices.
1 for each edge $e = (S_{i,i+1}, S_{j,j+1}) \in TF$ do
2     if $\frac{i+j}{2}$ is an even number then
3         Put the color of vertex $S_{\frac{i+j}{2},\frac{i+j}{2}+1}$ on the edge $e$;
4     else
5         Put the color of vertex $S_{\frac{i+j+2N}{2}\%2N,\frac{i+j+2N}{2}\%2N+1}$
          on the edge $e$;
```

rainbow-coloring algorithm.

Theorem 4: The coloring process is a rainbow-coloring process (i.e., after the coloring process, the colors on the N edges of each 2-factor are different).

Proof: Prove by contradiction. Suppose there exist two different edges $e_1 = (S_{i,i+1}, S_{j,j+1})$ and $e_2 = (S_{m,m+1}, S_{n,n+1})$ in the the 2-factor TF_d ($d \in [1, \frac{N-1}{2}]$) that have the same color. Since e_1 and e_2 are not the same, without loss of generality, let us assume $i \neq m$, $i > j$ and $m > n$.

Since e_1 and e_2 have the same color, they correspond to the same vertex in K_N. Without loss of generality, let us assume $\frac{i+j}{2}$ is an even number, then we have $\frac{i+j}{2} = \frac{m+n}{2}$ $(*)$.

Meanwhile, since e_1 and e_2 belong to the same 2-factor TF_d, we have $\frac{i-j}{2} = \frac{m-n}{2} = d$ $(**)$.

From $(*)$ and $(**)$ we have $i = m$, which is contradictory to $i \neq m$. Therefore, all the colors on the edges in the 2-factor TF_d are different. ■

Theorem 5: Put the $\frac{N-1}{2}$ 2-factor together (which forms the complete graph K_N), the colors on the $N-1$ edges incident to a vertex $S_{i,i+1}$ are different, and these colors are also different from the color of the vertex $S_{i,i+1}$.

Proof: Prove by contradiction. Suppose among the $N-1$ edges that are incident to the vertex $S_{i,i+1}$, there exist two edges that have the same color, and these two edges are $e_1 = (S_{i,i+1}, S_{j,j+1})$ and $e_2 = (S_{i,i+1}, S_{k,k+1})$. Because e_1 and e_2 have the same color, they correspond to the same vertex. Without loss of generality, let us assume $\frac{i+j}{2}$ is an even number. Then we have $\frac{i+j}{2} = \frac{i+k}{2} \Rightarrow j = k$, which is impossible since e_1 and e_2 are not the same. Therefore, the colors on the $N-1$ edges that are incident to the vertex $S_{i,i+1}$ are all different. ■

Theorem 6: Put the $\frac{N-1}{2}$ 2-factor together (which forms the complete graph K_N), the colors on the $N-1$ edges incident to a vertex $S_{i,i+1}$ are different from the color of the vertex $S_{i,i+1}$.

Proof: This is equivalent to prove that for an edge $e = (S_{i,i+1}, S_{j,j+1})$, the color of the edge e is different from the colors on both of its two vertices.

Without loss of generality, let us assume $\frac{i+j}{2}$ is an even number. If the color of e is the same as the color on the vertex

Algorithm 5: Single-phase CH sequence construction

```
Data: $N$ rendezvous channels $C_0, \cdots, C_{N-1}$, where $N$ is an odd number;
Result: $2N$ CH sequence $S_0, \cdots, S_{2N-1}$, each of which has $2N-1$ slots.
$S_0, \cdots, S_{2N-1} \leftarrow \phi$;
for $i \leftarrow 0$ to $N-1$ do
    $S_{2i} \leftarrow S_{2i} \cup (0, C_i)$;
    $S_{2i+1} \leftarrow S_{2i+1} \cup (0, C_i)$;
for $d \leftarrow 1$ to $\frac{N-1}{2}$ do
    $sl \leftarrow 1 + 4(d-1)$;
    for $a \leftarrow 0$ to $GCD(N, d) - 1$ do
        $i \leftarrow 2a$;
        for $c \leftarrow 0$ to $\frac{N}{GCD(N,d)} - 1$ do
            if $\frac{i+(i+2d)\%2N}{2}$ is an even number then
                $u \leftarrow \frac{i+(i+2d)\%2N}{4}$;
            else
                $u \leftarrow \frac{\frac{i+(i+2d)\%2N+2N}{2}\%2N}{2}$;
            if $c == 0$ then
                AddSlot($i, d, u, 0, 0, sl$);
                AddSlot($i, d, u, 0, 1, sl+1$);
                AddSlot($i, d, u, 1, 0, sl+2$);
                AddSlot($i, d, u, 1, 1, sl+3$);
            else if $c == 1$ then
                AddSlot($i, d, u, 1, 0, sl$);
                AddSlot($i, d, u, 0, 1, sl+1$);
                AddSlot($i, d, u, 1, 1, sl+2$);
                AddSlot($i, d, u, 0, 0, sl+3$);
            else if $c == 2$ then
                AddSlot($i, d, u, 1, 1, sl$);
                AddSlot($i, d, u, 0, 1, sl+1$);
                AddSlot($i, d, u, 0, 0, sl+2$);
                AddSlot($i, d, u, 1, 0, sl+3$);
            else if $c$ is an odd number then
                AddSlot($i, d, u, 0, 0, sl$);
                AddSlot($i, d, u, 0, 1, sl+1$);
                AddSlot($i, d, u, 1, 1, sl+2$);
                AddSlot($i, d, u, 1, 0, sl+3$);
            else
                AddSlot($i, d, u, 1, 1, sl$);
                AddSlot($i, d, u, 0, 1, sl+1$);
                AddSlot($i, d, u, 0, 0, sl+2$);
                AddSlot($i, d, u, 1, 0, sl+3$);
            $i \leftarrow (i + 2d)\%2N$;
return $S_0, \cdots, S_{2N-1}$;
```

$S_{i,i+1}$ or the color on the vertex $S_{j,j+1}$, we have $\frac{i+j}{2} = i$ or $\frac{i+j}{2} = j$, which implies $i = j$. This is impossible because $S_{i,i+1}$ and $S_{j,j+1}$ are two different vertices. Therefore, the color of edge e cannot be the same as the color on either of its vertices. ■

In our example, Figure 3(4a) and Figure 3(4b) show the rainbow-coloring for the two 2-factors of K_5.

Step #5 - expanding the rainbow-colored 2-factors of K_N back to 4-factors of K_{2N}: Here we expand each 2-factor of K_N back to a 4-factor of the $2N$-vertex complete graph K_{2N}. For each edge $e = (S_{i,i+1}, S_{j,j+1})$ in a 2-factor of K_N, we expand it to a *monochromatic complete bipartite graph* $K_{2,2}$:

Algorithm 6: Subfunction `AddSlot()` of Alg. 5

```
void AddSlot($i, d, u, a, b, sl$) {
    $S_{i+a} \leftarrow S_{i+a} \cup (sl, C_u)$ ;
    $S_{(i+2d)\%2N+b} \leftarrow S_{(i+2d)\%2N+b} \cup (sl, C_u)$;
}
```

$$MCB_{i,j} = (V_i + V_j, E_{ij}), \tag{8}$$

where V_i and V_j are the vertex pairs $\{S_i, S_{i+1}\}$ and $\{S_j, S_{j+1}\}$ in the original $2N$-vertex graph G, and E_{ij} is the edge set of $K_{2,2}$. Additionally, we give all the 4 edges in E_{ij} the same color as that on the edge $e = (S_{i,i+1}, S_{j,j+1})$ in K_N. After this process, every 2-factor in K_N is expanded to a 4-factor in K_{2N}. Since the $\frac{N-1}{2}$ 2-factors of K_N are different, we have obtained $\frac{N-1}{2}$ different 4-factors of K_{2N}.

Theorem 7: The $\frac{N-1}{2}$ different 4-factors of K_{2N} together with the first PRM obtained in the step #2 form the N-colored $2N$-vertex complete graph K_{2N} that has the properties of (1) to (5).

Proof: For the reasons that 1) the $\frac{N-1}{2}$ 4-factors are obtained by expanding each edge of the $\frac{N-1}{2}$ 2-factors of K_N to a complete bipartite graph $K_{2,2}$ and 2) each pair of unconnected vertices in a $K_{2,2}$ will be connected by an edge in the first PRM obtained in the step #1, we can conclude that the $\frac{N-1}{2}$ 4-factors together with the first PRM form a $2N$-vertex complete graph K_{2N} (i.e., the properties (1) and (3)).

For the reasons that 1) the coloring process of the 2-factors of K_N is a rainbow coloring process (i.e., Theorem 4) and 2) the color of each monochromatic complete bipartite graph (MCB) is taken from the corresponding edge of the 2-factor of K_N, we can conclude that all the N colors appear on K_{2N} (i.e., the property (2)).

For the reasons that 1) the colors on the $N-1$ edges incident to a vertex $S_{i,i+1}$ of K_N are different (i.e., Theorem 5), 2) these colors are also different from the color of the vertex $S_{i,i+1}$ (i.e., Theorem 6) and 3) the color of the vertex $S_{i,i+1}$ is the same as the color on the edge (S_i, S_{i+1}) of K_{2N} (i.e., the step #2), we can conclude that all the N colors appear on the $2N-1$ edges that are incident to the vertex S_i (which is also true for S_j)(i.e., the property (4)).

For the reasons that 1) the colors on the $N-1$ edges incident to a vertex $S_{i,i+1}$ of K_N are different (i.e., Theorem 5) and 2) each edge in K_N is expanded to two edges in K_{2N}, we can conclude that each of the $N-1$ colors appears twice on the edges that are incident to the vertex S_i (also true for S_{i+1}). Furthermore, for the reasons that 1) the color on the vertex $S_{i,i+1}$ is different from the previous $N-1$ colors (i.e., Theorem 6) and 2) the color of the vertex $S_{i,i+1}$ is the same as the color on the edge (S_i, S_{i+1}) of K_{2N} (i.e., the step #2), we can conclude that the color on $S_{i,i+1}$ appear once on the edges that are incident to the vertex S_i (also true for S_{i+1}). Therefore, the property (5) is satisfied. ■

1: $(\lambda_0[0], \lambda_1[0]), (\lambda_1[1], \lambda_2[0]), (\lambda_2[1], \lambda_0[1])$
2: $(\lambda_0[0], \lambda_1[1]), (\lambda_1[0], \lambda_2[1]), (\lambda_2[0], \lambda_0[1])$
3: $(\lambda_0[1], \lambda_1[0]), (\lambda_1[1], \lambda_2[1]), (\lambda_2[0], \lambda_0[0])$
4: $(\lambda_0[1], \lambda_1[1]), (\lambda_1[0], \lambda_2[0]), (\lambda_2[1], \lambda_0[0])$

Fig. 5. Dividing the edges of a $CMCB$ that contains 3 MCBs, i.e., $CMCB = \{\lambda_0, \lambda_1, \lambda_2\}$, into four groups such that all the edges in each group have no common vertex.

In our example, Figure 3(5a) and Figure 3(5b) show the two 4-factors of K_{10} that are converted from the two 2-factors of K_5. These two 4-factors and the first PRM obtained in the step #2 (i.e., Figure 3(2)) together form the complete graph K_{10} with the properties of (1) to (5) (Figure 4(a)).

Step #6 - decomposing the 4-factors of K_{2N} into perfect rainbow matchings in K_{2N}: Finally, we decompose each of the 4-factors of K_{2N} obtained in the step #5 into 4 different PRMs such that the properties (6) and (7) are satisfied.

In the previous step, each edge $(S_{i,i+1}, S_{j,j+1})$ in a 2-factor TF_d ($d \in [1, \frac{N-1}{2}]$) of K_N is expanded to a monochromatic complete bipartite graph $MCB_{i,j}$. Furthermore, recall that TF_d is either a Hamiltonian cycle of K_N (when $GCD(N,d) = 1$) or a set of $GCD(N,d)$ disjoint $\frac{N}{GCD(N,d)}$-cycles (when $GCD(N,d) \neq 1$), where $GCD(N,d)$ is the greatest common divisor of N and d. Therefore, the 4-factor FF_d of K_{2N}, which is expanded from the 2-factor TF_d of K_N, is a spanning graph of K_{2N} consisting of either one *chained monochromatic complete bipartite* graph (notated as $CMCB$) (when $GCD(N,d) = 1$) or a set of $GCD(N,d)$ disjoint $CMCB$s (when $GCD(N,d) \neq 1$).

Since a monochromatic complete bipartite graph $MCB_{i,j}$ connects two pairs of unconnected vertices $\{S_i, S_{i+1}\}$ and $\{S_j, S_{j+1}\}$, each $CMCB$ in the 4-factor FF_d ($d \in [1, \frac{N-1}{2}]$) of K_{2N} can be expressed as

$$< \lambda_0, \lambda_1, \cdots, \lambda_{n-1}, \lambda_0 >, \tag{9}$$

where $n = \frac{N}{GCD(N,d)}$ is the number of MCBs contained in the $CMCB$, and λ_p ($p \in [0, n-1]$) is the p-th unconnected vertex pair in the $CMCB$: $\{S_{2pd\%2N}, S_{(2pd+1)\%2N}\}$. For example, for the first 4-factor FF_1 of K_{10} shown in Figure 3(5a), we have $\lambda_0 = \{S_0, S_1\}$, $\lambda_1 = \{S_2, S_3\}$, $\lambda_2 = \{S_4, S_5\}$, $\lambda_3 = \{S_6, S_7\}$ and $\lambda_4 = \{S_8, S_9\}$. Meanwhile, for the second 4-factor FF_2 of K_{10} shown in Figure 3(5b), we have $\lambda_0 = \{S_0, S_1\}$, $\lambda_1 = \{S_4, S_5\}$, $\lambda_2 = \{S_8, S_9\}$, $\lambda_3 = \{S_2, S_3\}$ and $\lambda_4 = \{S_6, S_7\}$.

Given a $CMCB$ in a 4-factor expressed in formula (9), we divide the edges of the $CMCB$ into 4 groups as follows. For the 4 edges of each MCB, we put them into 4 groups respectively such that edges in the same group share no common vertex. Figure 5 and Figure 6 show the way we divide the edges. Figure 5 shows the case that the $CMCB$ has 3 MCBs, and Figure 6 shows the case that the $CMCB$ has more than 3 MCBs. In these two figures, $\lambda_p[0]$ and $\lambda_p[1]$ are the first and the second vertex of the vertex pair λ_p ($p \in [0,1]$) respectively.

If a 4-factor of K_{2N} contains one $CMCB$ (when $GCD(N,d) = 1$), the four edge groups obtained by using the dividing method shown in Figure 6 are the 4 different PRMs. If the 4-factor contains several disjoint $CMCB$s (when $GCD(N,d) \neq 1$), we put the i-th ($1 \leq i \leq 4$) edge group of each $CMCB$ into the same group to form a PRM. Therefore, each 4-factor of K_{2N} leads to 4 different PRMs, and the $\frac{N-1}{2}$ different 4-factors of K_{2N} produce $2N-2$ different PRMs. Adding the first PRM obtained in the step #1, we now have $2N-1$ different PRMs of K_{2N}. Each of these PRMs instructs the $2N$ CH sequences rendezvous in one of the $2N-1$ hopping slots of a hopping period.

Since the edges in the same PRM share no common vertex and the colors of the K_{2N}'s MCBs are different, the property (6) is satisfied. Meanwhile, since each edge is assigned to only one PRM, the $2N-1$ PRMs are different (i.e., the property (7) is satisfied).

In our 5-rendezvous channel network example (i.e., $N = 5$), using the dividing method in Figure 6, the first 4-factor of K_{10} (Figure 3(5a)) is decomposed into four different PRMs shown in Figure 3 (6a-1) to (6a-4), and the second 4-factor of K_{10} (Figure 3(5b)) is decomposed into another four different PRMs shown in Figure 3 (6b-1) to (6b-4). Based on the 9 PRMs (i.e., Figure 3(2), Figure 3 (6a-1) to (6a-4) and Figure 3 (6b-1) to (6b-4)), we construct the final $2N$ CH sequences shown in Figure 4(b).

3) The complete algorithm: The complete single-phase CH sequence construction algorithm is given in Algorithm 5 with its subfunction `AddSlot()` shown in Algorithm 6. Given a DSA network with N rendezvous channels, the algorithm outputs $2N$ CH sequences, each with $2N-1$ hopping slots, such that the following three conditions are satisfied. *First*, within the $2N-1$ hopping slots, every CH sequence meets with all the other $2N-1$ sequence each at a time in a hopping slot. *Second*, there are exactly two CH sequences hopping to the same rendezvous channel in a hopping slot. *Third*, in a CH sequence, each of the N rendezvous channels has the same probability to appear in the $2N-1$ hopping slots.

The algorithm essentially integrates the six intuitive steps described previously. Lines 2 to 4 of the algorithm schedule how the $2N$ CH sequences meet with each other in the slot-0, which is equivalent to forming the first PRM in step #1. Each iteration of the for-loop (lines 6 to 39) outputs rendezvous schedules for 4 hopping slots, which correspond to the 4 PRMs decomposed from a 4-factor of K_{2N}. Each iteration of the for loop (lines 8 to line 39) correspond to dividing the edges of a $CMCB$ into 4 groups (i.e., step #6). Lines 10 to 13 decides the channel to assigned, which correspond to the rainbow-coloring of 2-factors of K_N in step #4.

C. CH sequence execution

At the completion of constructing CH sequences by using either the two-phase CH sequence construction algorithm or the single-phase CH sequence construction algorithm, the newly joined node obtains a set of CH sequences, which are the same as those that any other nodes construct. Then the node synchronizes to the existing nodes using the global synchronization mechanism, and starts the channel hopping process described as follows. The node randomly selects a CH sequence to hop on. After hopping through all the slots, it performs the random CH sequence selection again and starts hopping on the newly chosen CH sequence. The node repeats

1: $(\lambda_0[0], \lambda_1[0]), (\lambda_1[1], \lambda_2[0]), (\lambda_2[1], \lambda_3[1]),$ $\boxed{(\lambda_3[0], \lambda_4[0]), (\lambda_4[1], \lambda_5[1]),}$ $\dots ,$ $\boxed{(\lambda_{n-2}[0], \lambda_{n-1}[0]), (\lambda_{n-1}[1], \lambda_0[1])}$

2: $(\lambda_0[0], \lambda_1[1]), (\lambda_1[0], \lambda_2[1]), (\lambda_2[0], \lambda_3[1]),$ $\boxed{(\lambda_3[0], \lambda_4[1]), (\lambda_4[0], \lambda_5[1]),}$ $\dots ,$ $\boxed{(\lambda_{n-2}[0], \lambda_{n-1}[1]), (\lambda_{n-1}[0], \lambda_0[1])}$

3: $(\lambda_0[1], \lambda_1[0]), (\lambda_1[1], \lambda_2[1]), (\lambda_2[0], \lambda_3[0]),$ $\boxed{(\lambda_3[1], \lambda_4[1]), (\lambda_4[0], \lambda_5[0]),}$ $\dots ,$ $\boxed{(\lambda_{n-2}[1], \lambda_{n-1}[1]), (\lambda_{n-1}[0], \lambda_0[0])}$

4: $(\lambda_0[1], \lambda_1[1]), (\lambda_1[0], \lambda_2[0]), (\lambda_2[1], \lambda_3[0]),$ $\boxed{(\lambda_3[1], \lambda_4[0]), (\lambda_4[1], \lambda_5[0]),}$ $\dots ,$ $\boxed{(\lambda_{n-2}[1], \lambda_{n-1}[0]), (\lambda_{n-1}[1], \lambda_0[0])}$

Fig. 6. Dividing the edges of a $CMCB = \{\lambda_0, \cdots, \lambda_{n-1}, \lambda_0\}$, where n is the number of MCBs in the $CMCB$ and n is greater than 3, into four groups such that all the edges in each group have no common vertex.

this process while it is idle. The reason for the node to re-select a CH sequence after a hopping period is to make sure that any pair of nodes are able to rendezvous in different rendezvous channels. Since the selection of CH sequence is random, the requirement of full utilization of rendezvous channels is satisfied. When a rendezvous channel's primary user appears, the nodes on that channel should yield using the channel, wait until a hopping slot, in which the rendezvous channel is available, is reached, and resume the hopping process.

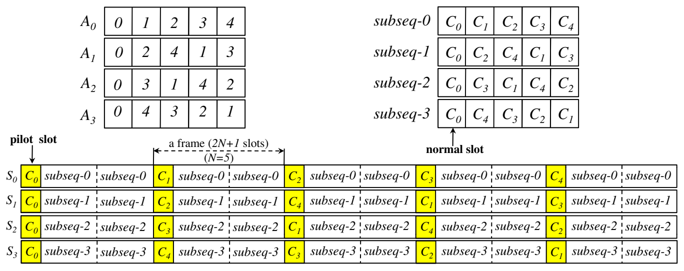
V. ASYNC-ETCH

Our study of the communication rendezvous so far is based on the assumption that there exists a global synchronization mechanism to synchronize the hopping processes of the nodes. In this section, we investigate the design of CH based communication rendezvous without leveraging the synchronization mechanism. Without synchronization, a pair of nodes wishing to communicate with each other start channel hopping at a random time. Consequently, their CH sequences are most probably misaligned and SYNC-ETCH cannot guarantee channel overlap for rendezvous. We develop an asynchronous scheme, ASYNC-ETCH, to address the issue.

ASYNC-ETCH follows the similar steps: the CH sequence construction and CH sequence execution. ASYNC-ETCH constructs the CH sequences in a similar fashion as SeqR [16] but employs a novel enhancement: it constructs multiple CH sequences rather than only one as in SeqR. The arrangement of having multiple sequences brings two benefits. First, multiple sequences reduce the chance that two nodes select the same CH sequence. As we will show later, it takes less time for two nodes to rendezvous when they select different sequences. Second, with multiple sequences, participating nodes have more chances to rendezvous with each other within a hopping period. We show that a pair of nodes using ASYNC-ETCH that select two different CH sequences are guaranteed to rendezvous in N slots (where N is the number of rendezvous channels) within a hopping period no matter how the hopping processes of the pair of nodes are misaligned.

A. An overview and an example

In a DSA network with five (N) rendezvous channel, the nodes first construct a set of four ($N-1$) CH sequences, S_0, S_1, S_2 and S_3, as shown in Fig. 7. As we can see from the lower part of the figure, each CH sequence consists of five (N) frames, each of which contains 11 ($2N+1$) slots: a pilot slot followed by two five-slot (N-slot) subsequences. The arrangement of the pilot slots is displayed in the the upper left part of the figure where pilot slot sequences A_0, A_1, A_2, A_3 are used in CH sequences S_0, S_1, S_2 and S_3, respectively. The arrangements for A_0 to A_3 are derived by the method of addition modulo the prime number five (N) with different addends from one to four respectively. The construction of



A0 0 1 2 3 4
A1 0 2 4 1 3
A2 0 3 1 4 2
A3 0 4 3 2 1
subseq-0 C0 C1 C2 C3 C4
subseq-1 C0 C2 C4 C1 C3
subseq-2 C0 C3 C1 C4 C2
subseq-3 C0 C4 C3 C2 C1
pilot slot
a frame (2N+1 slots) (N=5)
normal slot
S0 C0 subseq-0 subseq-0 C1 subseq-0 subseq-0 C2 subseq-0 subseq-0 C3 subseq-0 subseq-0 C4 subseq-0 subseq-0
S1 C0 subseq-1 subseq-1 C2 subseq-1 subseq-1 C4 subseq-1 subseq-1 C1 subseq-1 subseq-1 C3 subseq-1 subseq-1
S2 C0 subseq-2 subseq-2 C3 subseq-2 subseq-2 C1 subseq-2 subseq-2 C4 subseq-2 subseq-2 C2 subseq-2 subseq-2
S3 C0 subseq-3 subseq-3 C4 subseq-3 subseq-3 C3 subseq-3 subseq-3 C2 subseq-3 subseq-3 C1 subseq-3 subseq-3

Fig. 7. CH sequences of a DSA network with 5 rendezvous channels.

the four subsequences (shown in the upper right part of the figure) also follows the channel assignment order determined in A_0 to A_3. As we will prove later, the above CH sequence construction guarantees that any pair of nodes (selecting two different sequences) rendezvous in N slots within a hopping period regardless how much channel hopping misalignment between the two nodes. Each ASYNC-ETCH CH sequence has 55 slots ($N*(2N+1)$).

After finishing the CH sequences construction, the nodes start the same CH hopping execution as in SYNC-ETCH: each of them randomly selects a CH sequence to start, and randomly reselects another one to continue after hopping on the old one for a hopping period. By doing this, we ensure that any pair of nodes can rendezvous in different channels, which satisfies the requirement of full utilization of rendezvous channels. This arrangement also eliminates the unfairness that nodes selecting the same CH sequence have less chance to rendezvous than nodes selecting different CH sequences.

B. CH sequences construction

Algorithm 7 describes the construction of the $N-1$ CH sequences in ASYNC-ETCH. To ease our presentation, we assume the number of rendezvous channels, N, is a prime number. We hold the discussion of a general case (where N is not prime) till Section V-D.

Given N rendezvous channels, ASYNC-ETCH first derives $N-1$ integer sequences A_0 through A_{N-2} (which will be used as indices for later channel assignment) by applying addition modulo the prime number N (lines 1 to 4). Note that all the integer sequences are derived with different addends. In lines 5 to 7, the algorithm constructs $N-1$ CH subsequences, $subSeq_0$ to $subSeq_{N-2}$, whose channel indices are the same as the integer sequences A_0 through A_{N-2} respectively. Next, the algorithm constructs the CH sequence S_i ($0 \le i \le N-2$) by concatenating five frames of S_i together (line 8 to 15). Each frame of S_i consists of a *pilot slot* followed by a pair of $subSeq_i$. Slots in $subSeq_i$ are referred as *normal slots*. The channels in S_i's pilot slots, combined together, are exactly channels appearing in $subSeq_i$ in the same order. From Algorithm 7, it is easy to see that ASYNC-ETCH fulfills the requirement of even use of the rendezvous channels.

Algorithm 7: Async. CH sequence Construction

```
Data: $\mathcal{C} = \{C_0, \cdots, C_{N-1}\}$: $N$ rendezvous channels ($N$ is prime).
Result: $S_0, \cdots, S_{N-2}$: $N-1$ final CH sequences.
for $i \leftarrow 0$ to $N-2$ do
    $A_i[0] \leftarrow 0$;
    for $j \leftarrow 1$ to $N-1$ do
        $A_i[j] \leftarrow (A_i[0] + j(i+1)) \bmod N$;
for $i \leftarrow 0$ to $N-2$ do
    for $j \leftarrow 0$ to $N-1$ do
        $subSeq_i[j] \leftarrow C_{A_i[j]}$;
for $i \leftarrow 0$ to $N-2$ do
    $k \leftarrow 0$;
    for $j \leftarrow 0$ to $2N^2+N-1$ do
        if $j \bmod (2N+1) == 0$ then
            $S_i \leftarrow S_i \cup (j, subSeq_i[\frac{j}{2N+1}])$;   // pilot slot
        else
            $S_i \leftarrow S_i \cup (j, subSeq_i[k])$;   // normal slot
            $k \leftarrow (k+1) \bmod N$;
return $S_0, S_1, \cdots, S_{N-2}$;
```

C. Proof of rendezvous

In ASYNC-ETCH, the TTR between a pair of nodes is related to the fact that whether the two nodes select the same CH sequence or two different ones. Here we provide the theoretical analysis to determine the TTR performance in the above two situations. In particular, we prove that the two nodes have at least one overlapped CH slot within a hopping period in the former case, and they can rendezvous at least N times in the latter one.

Let us first rewrite the definition of *rotation closure property* from QCH [2] as follows.

Definition 1: Given a CH sequence S with p slots and a non-negative integer d, $\mathcal{R}(S,d) = \{(i, \mathcal{R}(S,d)[i]) \mid \mathcal{R}(S,d)[i] = S[(i+d) \bmod p]\}$ is called *a rotation of S with distance d.*

Definition 2: A CH sequence S with p slots is said to have the *rotation closure property with a degree of overlapping m* if $\forall d \in [0, p-1]$, $|S \cap \mathcal{R}(S,d)| \geq m$.

For instance, considering a CH sequence with three hopping slots, $S = \{(0, C_0), (1, C_0), (2, C_1)\}$, the two possible rotations are $\mathcal{R}(S,1) = \{(0,C_0),(1,C_1),(2,C_0)\}$ and $\mathcal{R}(S,2) = \{(0,C_1),(1,C_0),(2,C_0)\}$. It is obvious that S has the rotation closure property with a degree of overlapping 1.

Different from the prior work in SeqR [16], ASYNC-ETCH constructs multiple CH sequence rather than a single one. We provide the following definition to distinguish one CH sequence from another.

Definition 3: Two CH sequences, S_0 and S_1, each with p slots, are said be *different* if $\forall d \in [0, p-1]$, $S_1 \neq \mathcal{R}(S_0, d)$.

It is obvious that the $N-1$ CH sequences constructed by Algorithm 7 are different, since the subsequences, which are the building blocks of the CH sequences, are different.

We first analyze the case that two nodes select the same CH sequence.

Lemma 1: For two nodes periodically hopping on a CH sequence that has the closure property with a degree of overlapping m, they can rendezvous in at least $\frac{m}{2}$ slots within a hopping period no matter how their hopping processes are misaligned.

Proof: This lemma has been proved in QCH [2]. ■

Theorem 8: For two nodes that select the same CH sequence constructed by Algorithm 7, they can rendezvous in at least 1 slot within a hopping period no matter how their hopping processes are misaligned.

Proof: We need to prove that for any CH sequence S_i ($0 \leq i \leq N-2$) returned by Algorithm 7, S_i has the rotation closure property with a degree of overlapping 2, which combined with *Lemma* 1 can lead to this theorem. Specifically, we need to prove $\forall d \in [1, p-1]$, $\exists a \neq b \in [0, p-1]$ such that $S_i[a] = \mathcal{R}(S_i,d)[a]$ and $S_i[b] = \mathcal{R}(S_i,d)[b]$, where $p = 2N^2 + N$ is the number of slots of S_i.

If $d \bmod (2N+1) = 0$ (i.e., the 0-th slot of both $\mathcal{R}(S_i,d)$ and S_i are both pilot slots), then all $subSeq_i$ in both S_i and $\mathcal{R}(S_i,d)$ are aligned, there are $2N^2$ different overlappings.

If $d \bmod (2N+1) \neq 0$ (i.e., the 0-th slot in $\mathcal{R}(S_i,d)$ is a normal slot while the 0-th slot in S_i is a pilot slot), then we find the 2 overlappings as follows.

First, $\forall m, n \in [0, N-1]$ ($m \neq n$), we have $S_i[m(2N+1)] \neq S_i[n(2N+1)]$ (since the 0-th slot in S_i is a pilot slot) $\Rightarrow \bigcup S_i[p(2N+1)] = \{C_0, \cdots, C_{N-1}\}$, where $p = 0, \cdots, N-1$, and $\mathcal{R}(S_i,d)[m(2N+1)] = \mathcal{R}(S_i,d)[n(2N+1)] \in \{C_0, \cdots, C_{N-1}\}$ (since the 0-th slot in $\mathcal{R}(S_i,d)$ is a normal slot). Then there must exist a $p \in [0, N-1]$ such that $S_i[p(2N+1)] = \mathcal{R}(S_i,d)[p(2N+1)]$.

Second, for $k = 2N+1 - d \bmod (2N+1)$, the k-th slot in $\mathcal{R}(S_i,d)$ is a pilot slot while the k-th slot in S_i is a normal slot. Similar to the previous case, we can conclude that there exits an $p \in [0, N-1]$ such that $S_i[p(2N+1)+k] = \mathcal{R}(S_i,d)[p(2N+1)+k]$. ■

To determine the rendezvous performance when two nodes select two different CH sequences, we first give the definition of integer sequences derived by the method of addition modulo a prime number with different addends, and prove its overlap property.

Definition 4: Two integer sequences, $A = \{a_0, \cdots, a_{N-1}\}$ and $B = \{b_0, \cdots, b_{N-1}\}$ where N is a prime number, are said to be derived by the method of addition modulo the prime number N with different addends m and n if $a_i = (a_0 + im) \bmod N$, $b_i = (b_0 + in) \bmod N$, where $0 \leq a_0, b_0 \leq N-1$, $1 \leq i \leq N-1$ and $1 \leq m \neq n \leq N-1$.

Lemma 2: Given two integer sequences derived by the method of addition modulo a prime number with different addends m and n, $A = \{a_0, \cdots, a_{N-1}\}$ and $B = \{b_0, \cdots, b_{N-1}\}$, there must exist an integer $t \in [0, \cdots, N-1]$ such that $a_t = b_t$.

Proof: Prove by contradiction. Suppose $\forall t \in [0, \cdots, N-1]$, $a_t \neq b_t$. Construct a integers sequence $C = \{c_0, \cdots, c_{N-1}\}$, where $c_i = a_i - b_i$ $(0 \leq t \leq N-1)$. It is easy to see that $\forall c_i, c_j \in C$ $(0 \leq i \neq j \leq N-1)$, $c_i \neq c_j$, otherwise we can get $a_0 - b_0 + i(m-n) \equiv a_0 - b_0 + j(m-n) \pmod N \Rightarrow m - n$ is multiple times of N, which is impossible since $1 \leq m \neq n \leq N-1$. Because $a_t \neq b_t$ $\forall t \in [0, \cdots, N-1]$, C contains N different integers that are in the range of $[1, N-1]$, which is a contradiction. ■

Theorem 9: For two nodes that select two different CH sequence constructed by Algorithm 7, there must be at least N overlapping slots within a hopping period between the two CH sequences no matter how their hopping processes are misaligned.

Proof: Suppose S_i and S_j are two different CH sequences selected by the two nodes, we prove this theorem in the following two cases.

(1) The slot boundaries of S_i and S_j are aligned during the hopping processes of the two nodes. In this case, we have two further sub-cases as follows.

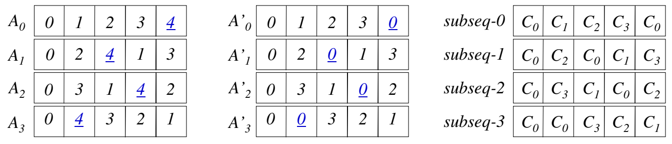
First, pilot slots in S_i overlap with pilot slots in S_j. In this case, all $subSeq_i$ in S_i exactly overlap with all $subSeq_j$ in S_j. Since integer sequences $\{A_i[0], \cdots, A_i[N-1]\}$ and $\{A_j[0], \cdots, A_j[N-1]\}$, which are the subscript sequences of $subSeq_i$ and $subSeq_j$ respectively, are derived by the method of addition modulo the prime number N with different addends, there exists one overlapping between a sub-sequence pair by *Lemma* 2. So there are $2N$ overlapping slots between S_i and S_j within a hopping period.

Second, pilot slots in S_i do not overlap with pilot slots in S_j. If the 0-th slot in S_i (a pilot slot) is aligned with the k-th $(0 \le k \le N-1)$ slot of the first $subSeq_j$ in a frame of S_j, then the first $subSeq_i$ in all the frames of S_i overlap with N contiguous normal slots in S_j. If the 0-th slot in S_i (a pilot slot) is aligned with the k-th $(0 \le k \le N-1)$ slot of the second $subSeq_j$ in a frame of S_j, then the first $subSeq_j$ in all the frames of S_j overlap with N contiguous normal slots in S_i. In either case, there exists at least one overlapping slot in each frame of both S_i and S_j because of *Lemma* 2 and the fact that the sequences of normal slots in S_i and S_j are developed by addition modulo prime the number N with different addends. So there are at least N overlapping slots between S_i and S_j within a hopping period.

(2) The slot boundaries of S_i and S_j are misaligned during the hopping processes of the two nodes. Suppose the first β $(0 < \beta < 1)$ portion of the 0-th slot in S_j overlaps with the l-th slot $(0 \le l \le 2N^2 + N)$ in S_i, then the rest $1-\beta$ portion of the 0-th slot in S_j overlaps with the l'-th slot in S_i, where $l' = (l+1) \bmod (2N^2 + N)$. Suppose the m-th slot in each frame of S_j is an overlapping slot if the boundaries of the 0-th slot in S_j and the l-th slot in S_i were aligned, and the n-th slot in each frame of S_j is an overlapping slot if the boundaries of the 0-th slot in S_j and the l'-th slot in S_i were aligned, then in each frame of S_j, S_j overlaps with S_i in the first β portion of the m-th slot and in the last $1-\beta$ portion of the n-th slot. In other words, there is at least one overlapping slot in each frame of both S_i and S_j. So there are at least N overlapping slots between S_i and S_j within a hopping period. ■

D. Additional discussion

Our previous analysis is based on the assumption that N is a prime number. To address the practical issue when N is not a prime number in a certain DSA network, we can make the following adjustment to easily remove the assumption. ASYNC-ETCH picks the smallest prime number that is greater than the number of rendezvous channels as the parameter N for Algorithm 7, and maps the excessive rendezvous channels



A_0	0	1	2	3	4
A_1	0	2	4	1	3
A_2	0	3	1	4	2
A_3	0	4	3	2	1

A'_0	0	1	2	3	0
A'_1	0	2	0	1	3
A'_2	0	3	1	0	2
A'_3	0	0	3	2	1

subseq-0	C_0	C_1	C_2	C_3	C_0
subseq-1	C_0	C_2	C_0	C_1	C_3
subseq-2	C_0	C_3	C_1	C_0	C_2
subseq-3	C_0	C_0	C_3	C_2	C_1

Fig. 8. ASYNC-ETCH CH sequences construction in a DSA network with 4 rendezvous channels.

down to the actual rendezvous channels. Fig. 8 demonstrates an example of ASYNC-ETCH CH sequences construction in a DSA network with 4 rendezvous channels C_0 to C_3. ASYNC-ETCH first constructs 4 integer sequences A_0 to A_3 using addition modulo a prime number 5 with addends 1 to 4 respectively. Then it converts the integer sequences A_i to A'_i $(0 \le i \le 3)$ by replacing number 4 with number 0 in A_i $(0 \le i \le 3)$. Then the ASYNC-ETCH CH sub-sequences will be constructed according to integer sequences A'_i $(0 \le i \le 3)$. The drawback of this method is that some rendezvous channels are assigned more times to the CH sequences. Therefore, for DSA networks using ASYNC-ETCH, we recommend to assign a prime number of channels for control information exchange.

VI. Comparisons

In this section, we theoretically compare ETCH with QCH [2] and SeqR [16], which are two existing CH based solutions for communication rendezvous in DSA networks.

In QCH, three versions of communication rendezvous protocols are designed. M-QCH and L-QCH are two synchronous versions that assume clocks are synchronized between nodes, and A-QCH is the asynchronous version that is used without such an assumption. The design goal of M-QCH is to minimize time-to-rendezvous between two CH sequences, while L-QCH's goal is to minimize the number of nodes that rendezvous in the same channel. SeqR is a DSA network communication rendezvous protocol without assuming global clock synchronization. SeqR does not have a synchronous version. We divide the comparisons into two group. In the first group, we compare SYNC-ETCH with M-QCH and L-QCH, all of which assume the existence of global clock synchronization. In the second group, we compare three asynchronous protocols: ASYNC-ETCH, A-QCH and SeqR.

We compare the two groups of communication rendezvous protocols on the three metrics introduced in section III-B: average rendezvous channel load, average TTR and rendezvous channels utilization ratio. Note that the choice of the CH sequence construction algorithm in the SYNC-ETCH protocol, i.e., the two-phase algorithm or the single-phase algorithm, makes no difference on the protocol's theoretical performances on the three metrics, because we do not consider the impacts of the appearances of primary users in these theoretical comparisons. We will evaluate how the appearances of primary users have impacts on the performances of the SYNC-ETCH protocol using different CH construction algorithms later in Section VII-B.

Table I summarizes the comparison results, where N is the number of rendezvous channels of the DSA network. In the synchronous protocols group, we pick parameters for L-QCH such that it produces the same number of CH sequences

TABLE I
COMPARISONS BETWEEN COMMUNICATION RENDEZVOUS PROTOCOLS

	Avg. Rend. channel load	Average TTR	Rend. channels utilization ratio
M-QCH	$\frac{2}{3}$	$\frac{3}{2}$	$\frac{1}{N}$
L-QCH	$\approx \frac{1}{\sqrt{2N-1}}$	$\frac{2N-1}{2}$	$\frac{1}{N}$
SYNC-ETCH	$\frac{1}{N}$	$\frac{2N-1}{2}$	1
A-QCH	$\frac{1}{2}$	$\geq \frac{9}{2}$	N/A
SeqR	$\frac{1}{N}$	$\frac{N^2+N}{2}$	N/A
ASYNC-ETCH	$\frac{1}{N}$	$\frac{2N^2+N}{N-1} \approx 2N$	N/A

as SYNC-ETCH for the purpose of fair comparison. SYNC-ETCH outperforms M-QCH and L-QCH on the metrics of average rendezvous channel load and rendezvous channels utilization ratio, because in every hopping slot it efficiently utilizes all rendezvous channels in establishing control channels, while there is only one channel can be used as control channel in each hopping slot with M-QCH and L-QCH. Thus theoretically, SYNC-ETCH experiences less traffic collisions and achieves higher throughput than QCH. For the metric of average TTR, M-QCH achieves the best theoretical performance. However, it has a very large average load on each rendezvous channel ($\frac{2}{3}$ of all the network nodes use the same rendezvous channel), which will cause a high probability of traffic collisions and further make the time-to-rendezvous performance of M-QCH worse than its theoretical value in practice.

In the asynchronous protocols group, A-QCH has the worst performance in terms of average rendezvous channel load, because it only ensures two of the rendezvous channels can be used as control channels while both ASYNC-ETCH and SeqR utilize all the rendezvous channels in control channel establishment. Moreover, A-QCH cannot provide a bounded TTR. SeqR, which constructs only one CH sequence, can only guarantee one overlapping slot in a hopping period. So the average TTR for SeqR is half of the number of slots in the CH sequence (i.e., $\frac{N^2+N}{2}$). For ASYNC-ETCH's performance on the metric of average TTR, we make the following analysis: we proved in section V-C that for the cases that when two nodes select the same CH sequence and when they select two different CH sequences, they are respectively guaranteed to meet in at least 1 slot and at least N slot within a hopping period. Since ASYNC-ETCH generates $N-1$ different CH sequences and the CH sequence selection is random, on average there are $\frac{1}{N-1} + \frac{(N-2)N}{N-1} = N-1$ guaranteed overlapping slots in a hopping period. So the average TTR for ASYNC-ETCH is $\frac{2N^2+N}{N-1} \approx 2N$.

VII. PERFORMANCE EVALUATION

We evaluate ETCH's performance by simulation experiments. In section VII-A, we compare ETCH with the existing CH based communication rendezvous protocols. In Section VII-B, we compare the two algorithms of SYNC-ETCH for CH sequence construction, i.e., the two-phase algorithm and the single-phase algorithm.

A. Comparing ETCH to the existing CH based communication rendezvous protocols

1) Methodology: We evaluate ETCH by comparing it to QCH and SeqR in the `ns-2` simulator. We divide the evaluation into two portions based on the assumption about the existence of global clock synchronization. In section VII-A2, we compare the performances of SYNC-ETCH (using the two-phase algorithm for CH sequence construction), M-QCH and L-QCH. In section VII-A3, we compare the performances of ASYNC-ETCH with A-QCH and SeqR.

In the evaluation, we modify the `ns-2` simulator to make it be able to perform multi-channel wireless communication simulations based on the Hyacinth project [21]. In our simulations, there are a varying number of nodes in a $500m \times 500m$ area, where each of the nodes is in all other nodes' communication ranges. The length of a hopping slot is set to 100 ms. We establish Constant Bit Rate (CBR) flows, where the packet size is set to 800 bytes and the packet rate is 125 packets/sec, from each node to all other nodes. These flows are started and stopped randomly during the simulation such that there is no more than one flow from the same node is activated simultaneously (because there is only one transceiver equipped with each node). Hyacinth's manual routing protocol is used in routing packets between the nodes. We disable the RTS/CTS function in the simulator, and rely on the retransmission mechanism to deal with packet collisions. In the simulations, the DSA network has 5 rendezvous channels (i.e., $N = 5$), each of which can possibly be used by the primary user. To simplify the simulation, we suppose all the secondary users are within the communication range of the primary user. The appearances of the primary user is simulated as follows. We first decide whether the primary user shows or not by flipping a coin. If the primary user appears, we randomly disable a rendezvous channel for a random period of time. Otherwise all the rendezvous channels are made to be available to the nodes also for a random period of time. We repeat this process during the entire simulation.

2) Synchronous communication rendezvous protocols: We conduct two simulation experiments to study the performances of the synchronous protocols on traffic throughput and actual time-to-rendezvous (TTR). In each experiment, we run the simulation for ten rounds with different number of secondary users (from 5 to 50 with a step length of 5) in each round.

Fig. 9 shows the traffic throughput performances of the three synchronous protocols. Part (a) of this figure shows the actual throughput while part (b) illustrates the improvement ratio curves of SYNC-ETCH over L-QCH and M-QCH. SYNC-ETCH has a lower throughput than L-QCH and M-QCH when there are 5 secondary users in the network. This is because in CH sequences of L-QCH and M-QCH, rendezvous channels are randomly assigned to those non-frame-channel-slots, which may give a pair of nodes using L-QCH or M-QCH extra slots to rendezvous in other than the frame-channel-slot. And this is also because there are no or little collisions in this case. However, when the number of secondary users is equal or greater than 10, SYNC-ETCH achieves higher traffic throughput than L-QCH and M-QCH, especially when the

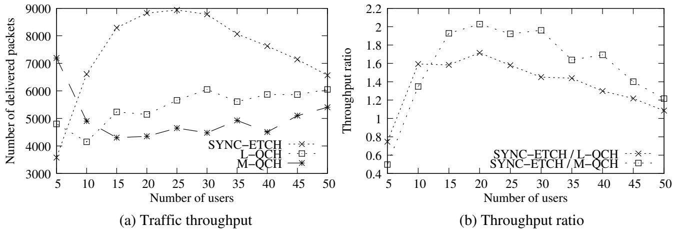


Fig. 9. Throughput performances of the synchronous protocols.

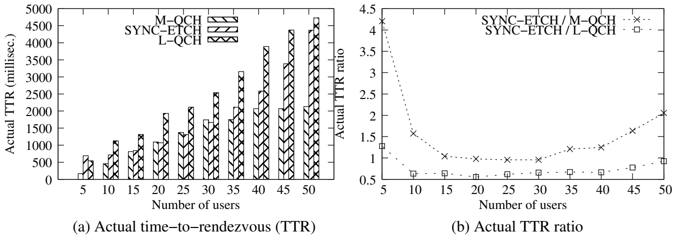


Fig. 10. TTR performances of the synchronous protocols.

nodes-channels ratio is in the range of 3 to 6 (i.e. when there are 15 to 30 nodes in the DSA network). In this case, traffic collision dominates the factors that influence the throughput performance. With both L-QCH and M-QCH, nodes are always compete for one rendezvous channel as control channel leaving all other rendezvous channels unused in a hopping frame, which causes a high probability of collisions when the nodes-channels ratio is bigger than 1. On the contrary, SYNC-ETCH schedules rendezvous among its CH sequences such that all the rendezvous channels can be utilized in every hopping slot. This approach greatly reduces traffic collisions and hence increases throughput. Furthermore, it can be also noticed in Fig 9 that the throughput performance of the three synchronous protocols converges as the nodes-channels ratio approaches 10. This is because collisions dominate traffics in each rendezvous channel with all the synchronous protocols. In this case, it is suggested to assign more rendezvous channels to accommodate such a high number of secondary users.

Fig. 10 part (a) shows the TTR performances of the three synchronous protocols, and part (b) demonstrates the TTR ratios of SYNC-ETCH over L-QCH and M-QCH. The TTRs of the three protocols increase as the number of secondary users grows because of the increasing traffic collisions. Although M-QCH achieves the best TTR performance among the three as analyzed in section VI, it does not get the theoretical TTR performance boost over SYNC-ETCH as analyzed in Section VI. Theoretically, M-QCH performs 3 times better than SYNC-ETCH in TTR, because it has an average TTR of 1.5 while SYNC-ETCH's value is 4.5. However, the simulation results shows that SYNC-ETCH's actual TTR is only 1.5 times of M-QCH's actual TTR on average. The reason of M-QCH's TTR performance degradation in the simulation experiment is because the nodes using M-QCH experience more severe traffic collisions that those using SYNC-ETCH.

From the above two simulations it can be seen that SYNC-ETCH achieves the best balance between traffic throughput and TTR among the three synchronous protocols.

3) Asynchronous communication rendezvous protocols: In this subsection, we compare the throughput and the TTR performances between the three asynchronous protocols: ASYNC-ETCH, A-QCH and SeqR.

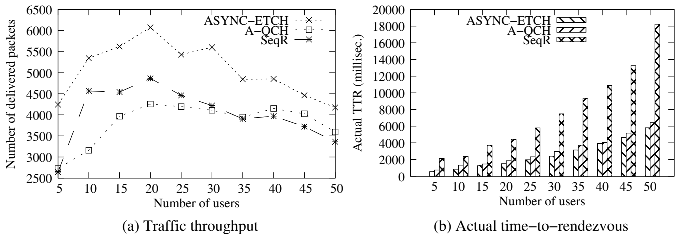


Fig. 11. Throughput and TTR of the asynchronous protocols.

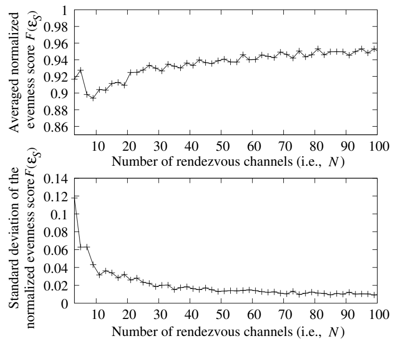


Fig. 12. Channel appearance evenness score of the two-phase CH sequence construction algorithm. (The evenness score of the single-phase algorithm is always 1).

Fig. 11 shows the performances of the three asynchronous protocols. In Fig. 11 part (a), the traffic throughput performances are shown. ASYNC-ETCH performs constantly better than the other two protocols in this metric. This is because ASYNC-ETCH is able to utilize all the rendezvous channels as control channels while A-QCH uses only two of them. Meanwhile, ASYNC-ETCH improves on SeqR such that it achieves a shorter average TTR, which contributes to the throughput performance boost over SeqR. Fig. 11 part (b) shows the actual TTR performances of the three protocols. It is not surprised that ASYNC-ETCH performance better than SeqR, because ASYNC-ETCH's average TTR is shorter than that of SeqR (see Table I for details). For A-QCH, we construct CH sequences such that they have an average TTR of 4.5, which is the best that A-QCH is able to achieve. Even so, ASYNC-ETCH still performs better than A-QCH.

B. Comparing the two algorithms for CH sequence construction in SYNC-ETCH

In the SYNC-ETCH protocol, we have proposed two algorithms for CH sequence construction. The two-phase algorithm can be applicable to DSA networks with an arbitrary number of rendezvous channels. However, it is unable to guarantee the even use of rendezvous channels requirement. The single-phase algorithm improves on its two-phase counterpart in that it guarantees, under the premise that N (i.e., the number of the rendezvous channels) is an odd number, all the rendezvous channels appear in each constructed CH sequence with the same probability.

To quantize how even the N rendezvous channels (i.e., $C_0, \cdots, C_{N-1}$) appear in a CH sequence S, we define the "evenness score" of S regarding rendezvous channel appear-

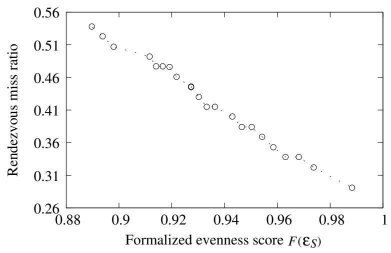


Fig. 13. Rendezvous miss ratio vs. channel appearance evenness score.

ance probability as

$$\varepsilon_S = \sqrt{\frac{\sum_{i=0}^{N-1}(a_i - \frac{|S|}{N})^2}{N}},$$

where $|S|$ is the number of hopping slots of S, and a_i is the number of hopping slots of S in which channel C_i appears. We further convert ε_S into a normalized score $N(\varepsilon_S)$, which is the range of $[0,1]$ and can be expresses as

$$N(\varepsilon_S) = 1 - \frac{\varepsilon_S - \varepsilon_{best}}{\varepsilon_{worst} - \varepsilon_{best}}.$$

In $N(\varepsilon_S)$, ε_{best} and ε_{worst} are the evenness scores of the best case and the worst case of fulfilling the even use of rendezvous channels requirement respectively. In the best case, each of the N rendezvous channels appears in S with the same the same probability, while in the worst case, a single channel appears in all the hopping slots of S. For instance, with the SYNC-ETCH protocol where there are $2N-1$ hopping slots in a CH sequence, the best case that a CH sequence S satisfies the even use of rendezvous channels requirement is that a rendezvous channels appears once in S while each of the remaining $N-1$ channels appears twice in S. The evenness score of the best case is calculated as $\varepsilon_{best} = \sqrt{\frac{(1-\frac{2N-1}{N})^2+(N-1)(2-\frac{2N-1}{N})^2}{N}}$. In the worst case, all the $2N-1$ slots is assigned with the same CH sequence. Accordingly, the evenness score of the worst case is calculated as $\varepsilon_{worst} = \sqrt{\frac{((2N-1)-\frac{2N-1}{N})^2+(N-1)(0-\frac{2N-1}{N})^2}{N}}$.

Low normalized evenness score of a CH sequence S indicates that S uses one or several rendezvous channels more than the remaining channels, which causes the nodes selecting S to have higher probability to experience communication outages if the primary users of those heavily relied channels show up. In SYNC-ETCH, every CH sequence constructed by the single-phase algorithm has a normalized evenness score of 1, which is the optimal case of fulfilling the even use of rendezvous channels requirement. To evaluate how well the two-phase algorithm satisfies this requirement, we calculate the average value and the corresponding standard deviation of the evenness scores of the $2N$ CH sequences constructed by the two-phase algorithm. Figure 12 shows the results of the cases where the value of N ranges from 3 to 99. The top graph of Figure 12 plots the average value of the evenness scores, and the bottom graph plots the corresponding standard deviations. We can see from the results that the two-phase algorithm still achieves an average normalized evenness score that is larger than 0.9 when N is greater than 10, and that the averaged score increases as N increases.

We further perform an experiment to evaluate how the normalized evenness scores of CH sequences affect the performances of the communication rendezvous protocol. In the experiment, we let a node A that is stick to a fixed CH sequence S_i rendezvous with another node B for $2N-1$ times, where the node B selects a different CH sequence S_j $(j \neq i)$ at each time. We disable γ $(0 < \gamma < 1)$ of the rendezvous channels that are used most frequently in S_i. The node A fails to rendezvous with the node B at a time if the overlapping channel between S_i and S_j is disable. We then calculate "rendezvous miss ratio" of the CH sequence S_i as the ratio between the number of times when a rendezvous attempt fails and the total number of rendezvous attempts (i.e., $2N-1$). Figure 13 (b) plots the relationship between the normalized evenness score and the rendezvous miss ratio of a CH sequence constructed by the two-phase algorithm S when $N = 33$ and $\gamma = 0.3$. Under the same settings, the rendezvous miss ratio of a CH sequence constructed by the single-phase algorithm is 0.27.

VIII. Conclusion

We have presented ETCH, efficient channel hopping based communication rendezvous protocols for DSA networks. ETCH protocols include SYNC-ETCH and ASYNC-ETCH. SYNC-ETCH, which assumes global clock synchronization, efficiently utilizes all the rendezvous channels in establishing control channels all the time. ASYNC-ETCH is able to make a pair of nodes rendezvous without being synchronized. Using a combination of theoretical analysis and simulations, we show that ETCH protocols perform better than the existing solutions for communication rendezvous in DSA networks.

References

[1] J. Zhao, H. Zheng, and G. H. Yang, "Distributed coordination in dynamic spectrum allocation networks," in *IEEE DySPAN*, December 2005.

[2] K. Bian, J. Park, and R. Chen, "A quorum-based framework for establishing control channels in dynamic spectrum access networks," in *ACM Mobicom*, June 2009.

[3] Y. Zhang, Q. Li, G. Yu, and B. Wang, "ETCH: Efficient Channel Hopping for communication rendezvous in dynamic spectrum access networks," in *INFOCOM*, 2011.

[4] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad hoc wireless networks," in *ACM Mobicom*, September 2004.

[5] A. Tzamaloukas and J. J. Garcia-Luna-Aceves, "Channel-Hopping Multiple Access," in *IEEE ICC 2000*, 2000.

[6] 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 JOURNAL (ELSEVIER)*, 2006.

[7] A. Sahai, N. Hoven, and R. Tandra, "Some Fundamental Limits on Cognitive Radio," in *Allerton Conference on Communication, Control, and Computing*, 2004.

[8] D. Cabric, S. M. Mishra, and R. W. Brodersen, "Implementation issues in spectrum sensing for cognitive radios," in *Asilomar Conference on Signals, Systems, and Computers*, 2004.

[9] A. Fehske, J. Gaeddert, and J. Reed, "A new approach to signal classification using spectral correlation and neural networks," in *IEEE DySPAN*, 2005.

[10] A. Ghasemi and E. Sousa, "Collaborative spectrum sensing for opportunistic access in fading environments," in *IEEE DySPAN*, 2005.

[11] S. Shankar, "Spectrum Agile Radios: Utilization and Sensing Architectures," in *IEEE DySPAN*, 2005.

[12] B. Wild and K. Ramchandran, "Detecting Primary Receivers for Cognitive Radio Applications," in *IEEE DySPAN*, 2005.

[13] H. Zheng and L.Cao, "Device-centric Spectrum Management," in *IEEE DySPAN*, 2005.

[14] V. Brik, E. Rozner, S. Banarjee, and P. Bahl, “DSAP: a protocol for coordinated spectrum access,” in *IEEE DySPAN*, 2005.

[15] L. Ma, X. Han, and C. Shen, “Dynamic open spectrum sharing MAC protocol for wireless ad hoc networks,” in *IEEE DySPAN*, 2005.

[16] L. A. DaSilva and I. Guerreiro, “Sequence-Based Rendezvous for Dynamic Spectrum Access,” in *IEEE DySPAN*, October 2008.

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

[18] *Maxim 2.4GHz 802.11b Zero-IF Transceivers*, Maxim Integrated Products. [Online]. Available: http://pdfserv.maxim-ic.com/en/ds/MAX2820-MAX2821.pdf

[19] Wikipedia, *Matching (graph theory)*, http://en.wikipedia.org/wiki/Matching_(graph_theory).

[20] T. D. LeSaulnier, C. Stocker, P. S. Wenger, and D. B. West, “Rainbow Matching in Edge-Colored Graphs,” *Electr. J. Comb.*, 2010.

[21] “Hyacinth: An IEEE 802.11-based Multi-channel Wireless Mesh Network,” *http://www.ecsl.cs.sunysb.edu/multichannel/*.

Appendix A

Here we give a more general example about the single-phase CH sequence construction process. In this example, we consider a DSA network with 9 rendezvous channels (i.e., $N = 9$), where eventually 18 CH sequences are constructed, each with 17 hopping slots.

Figure 14 shows the initial state of the graph G, the step #1 and #2 in the intuitive description of the single-phase CH sequence construction algorithm. (a) shows the initial state of the graph G, where the 18 vertices $S_0, \cdots, S_{17}$ correspond to the 18 CH sequences to construct. It also shows that each of the 9 rendezvous channels, $C_0, \cdots, C_8$, has been assigned a unique color. (b) shows the first perfect rainbow matching (PRM) of 18-vertex complete graph K_{18} obtained in the step #1. The first $PRMs$ tells how the 18 CH sequences rendezvous with each other in slot-0 by fully utilizing all the 9 rendezvous channels. (c) and (d) show how the 18-vertex graph G is shrank to a 9-vertex complete graph K_9 in the step #2 according to the first PRM obtained in the step #1: Two vertices S_i and S_{i+1} are combined to a new vertex $S_{i,i+1}$ if vertex S_i and vertex S_{i+1} are adjacent in the first PRM, and the new vertex $S_{i,i+1}$ is assigned the color as that of the edge connecting S_i and S_{i+1} in the first PRM (Figure (c)). Then the 9-vertex complete graph K_9 is formed on the 9 new vertex (Figure (d)).

Figure 15 shows the result of the step #3 in the intuitive description of the single-phase CH sequence construction algorithm - decomposing K_N into $\frac{N-1}{2}$ different 2-factors. In this figure, (a), (b) (c) and (d) are four different 2-factors of K_9 that are obtained by using vertex subscript difference (i.e., the parameter d in the description) 1, 2, 3 and 4 respectively. Since 1, 2, and 4 are all coprime with N, (a), (b) and (d) are three different Hamiltonian cycles of K_9. (c) is a spanning subgraph of K_9 containing a set of 3 3-cycles.

Figure 16 shows the result of the step #4 in the intuitive description of the single-phase CH sequence construction algorithm - rainbow-coloring the $\frac{N-1}{2}$ 2-factors of K_9. In this step, each of the four 2-factors of K_9 shown in Figure 15 is rainbow-colored using Algorithm 4.

Figure 17 shows the result of the step #5 in the intuitive description of the single-phase CH sequence construction algorithm - expanding the $\frac{N-1}{2}$ rainbow-colored 4-factors of k_9 back to the corresponding 4-factors in K_{18}. In this step, each of the four rainbow-colored 2-factors of K_9 shown in Figure 16 are expanded back to a corresponding 4-factor in K_{18} by converting each connected vertex pair in a 2-factor of K_9 to a monochromatic complete bipartite graph $K_{2,2}$ (the color is the same as in the original edge in K_9).

Figure 18 to Figure 21 show the result of the step #5 in the intuitive description of the single-phase CH sequence construction algorithm - decomposing the $\frac{N-1}{2}$ 4-factors of K_{2N} into $2N-2$ perfect rainbow matchings. The total 16 PRMs shown in these four figures tell how the 18 CH sequences rendezvous with each other in slot-1 to slot-16 respectively by fully utilizing all the 9 rendezvous channels in each hopping slot.

Figure 22 shows the final 18-vertex complete graph K_{18} and the final 18 CH-sequences. The 16 PRMs shown in Figure 18 to Figure 21 and the first PRM (shown in Figure 22 (a)) together form the final 18-vertex complete graph K_{18} (shown in Figure 22 (b)). The final 18 CH sequences (shown in Figure 22 (c)) is constructed based on the 17 PRMs, each of which tells how the 18 CH sequences rendezvous with each other in a different hopping slot by fully utilizing all the 9 rendezvous channels.

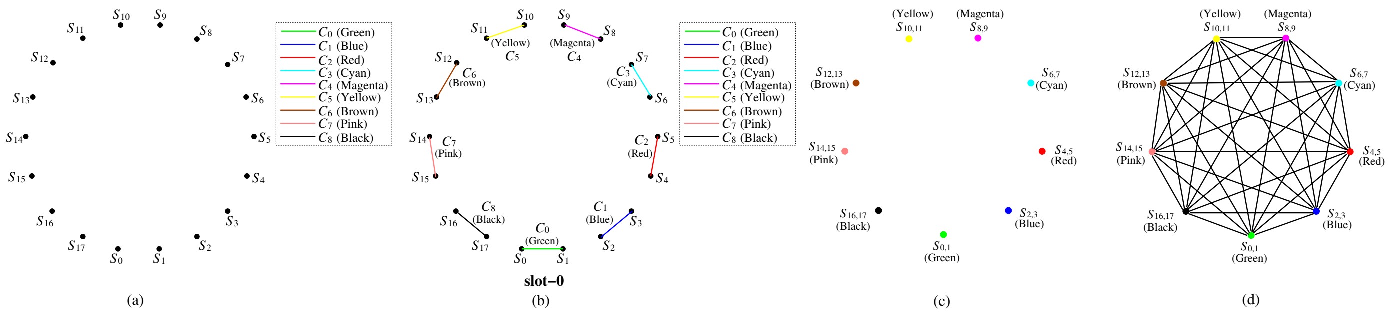


Fig. 14. The initial state of the graph G, the steps #1 and #2 in the intuitive description of the single-phase CH sequence construction algorithm in a DSA network with 9 rendezvous channels (i.e., $N = 9$). (a) is the initial state of the graph G; (b) shows the first PRM of the complete graph K_{18} obtained in the step # 1; The first $PRMs$ tells how the 18 CH sequences rendezvous with each other in slot-0 by fully utilizing all the 9 rendezvous channels; (c) and (d) show how the 18-vertex graph G so far is shrank to a 9-vertex complete graph K_9 in step #2.

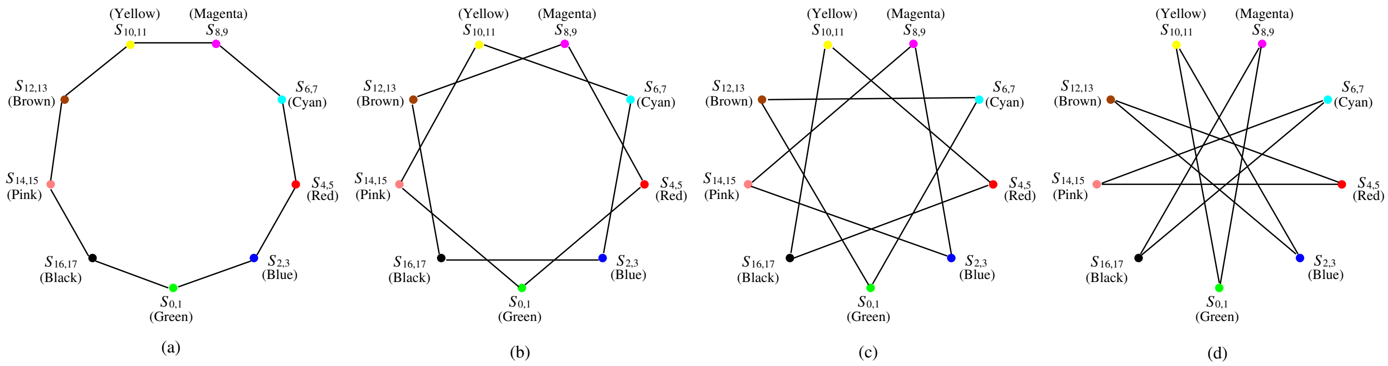


Fig. 15. Step #3 in the intuitive description of the single-phase CH sequence construction algorithm. In this step, the 9-vertex complete graph K_9 shown in Figure 14 (d) is decomposed into four different 2-factors of $K9$ shown in (a) to (d) respectively. (a), (b) and (d) are three different Hamiltonian cycles of K_9. (c) is a spanning subgraph of K_9 containing a set of 3 3-cycles.

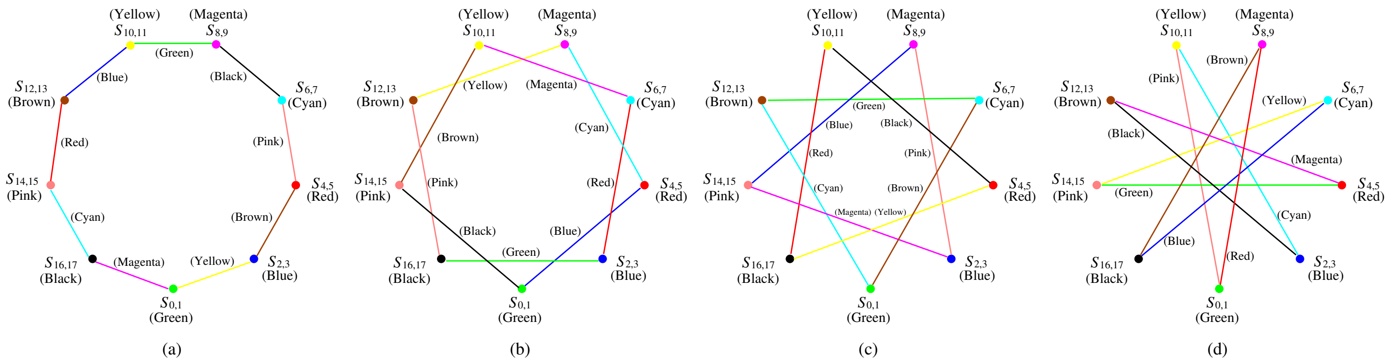


Fig. 16. Step #4 in the intuitive description of the single-phase CH sequence construction algorithm. In this step, each of the four 2-factors of K_9 shown in Figure 15 is rainbow-colored using Algorithm 4.

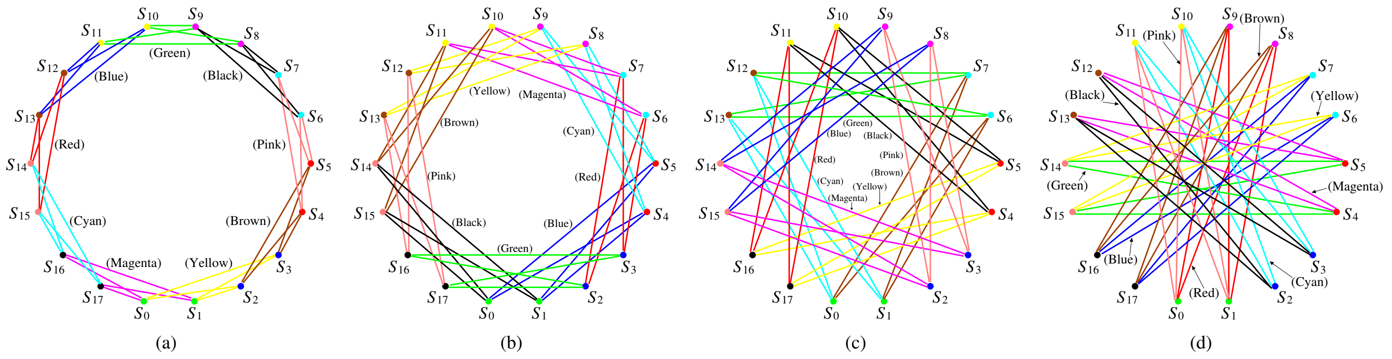


Fig. 17. Step #5 in the intuitive description of the single-phase CH sequence construction algorithm. In this step, each of the four rainbow-colored 2-factors of K_9 shown in Figure 16 are expanded back to a corresponding 4-factor in K_{18}.

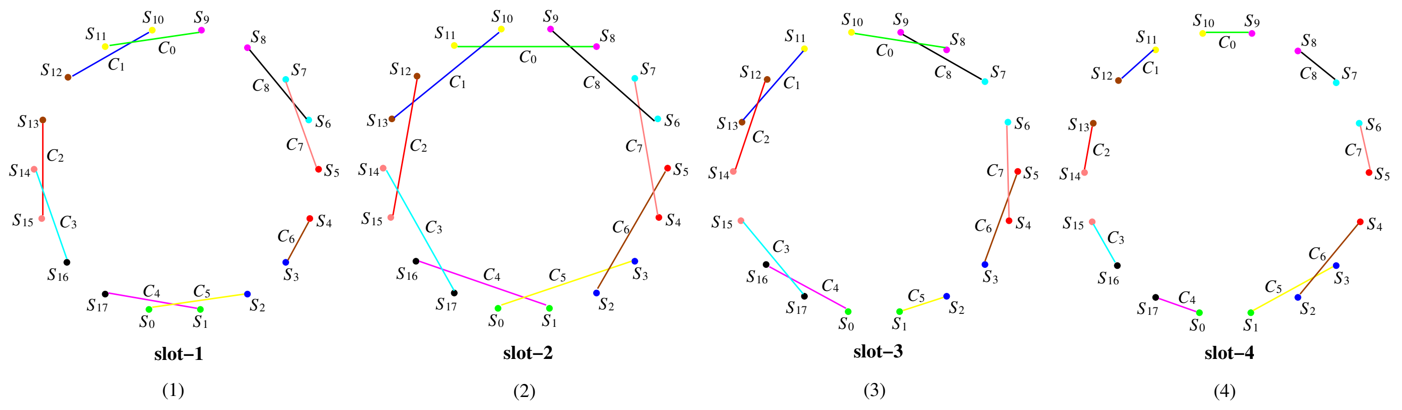


Fig. 18. The four different perfect rainbow matchings decomposed from the 4-factor of K_{18} shown in Figure 17 (a). These four PRMs tell how the 18 CH sequences rendezvous with each other in slot-1 to slot-4 respectively by fully utilizing all the 9 rendezvous channels in each hopping slot.

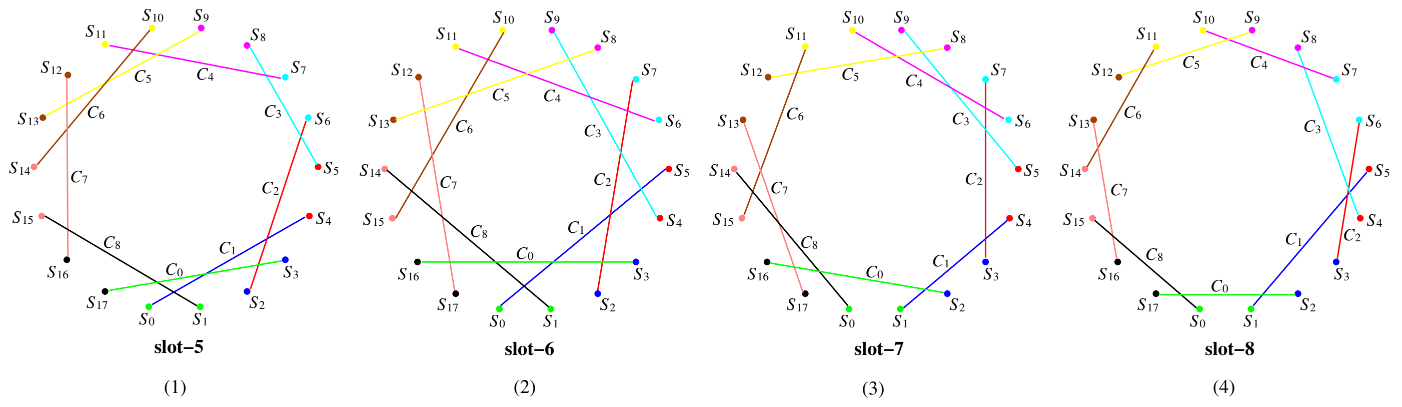


Fig. 19. The four different perfect rainbow matchings decomposed from the 4-factor of K_{18} shown in Figure 17 (b). These four PRMs tell how the 18 CH sequences rendezvous with each other in slot-5 to slot-8 respectively by fully utilizing all the 9 rendezvous channels in each hopping slot.

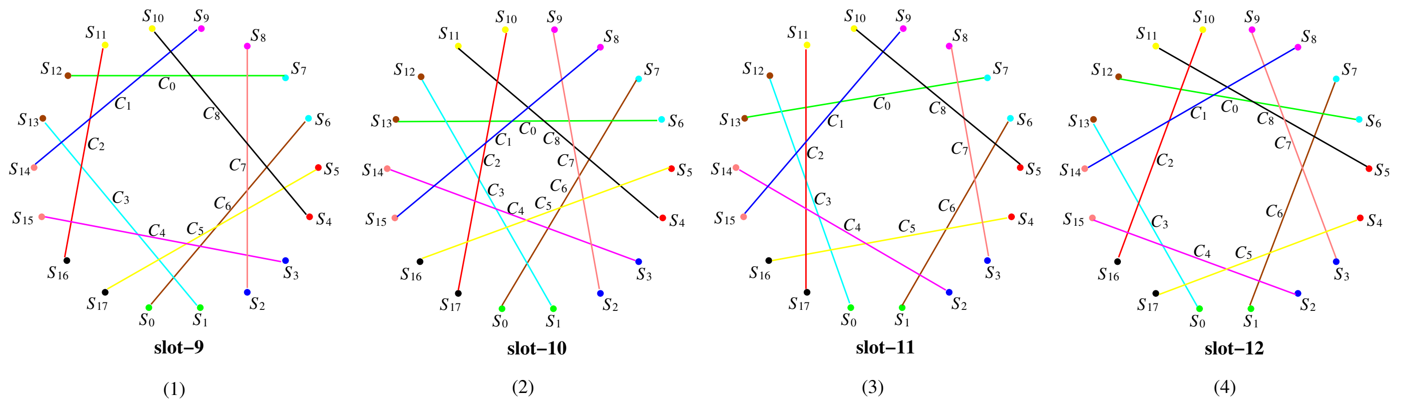
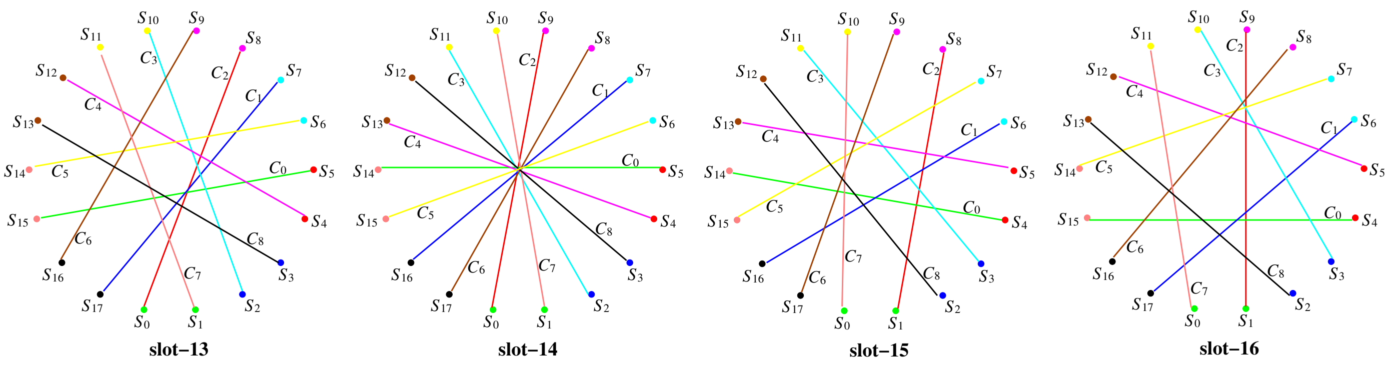
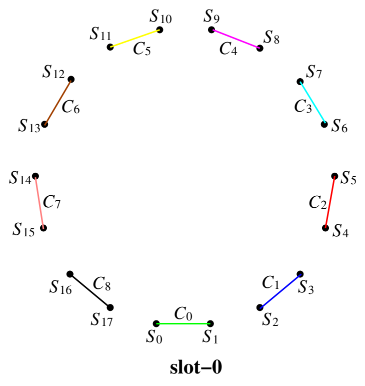


Fig. 20. The four different perfect rainbow matchings decomposed from the 4-factor of K_{18} shown in Figure 17 (c). These four PRMs tell how the 18 CH sequences rendezvous with each other in slot-9 to slot-12 respectively by fully utilizing all the 9 rendezvous channels in each hopping slot .

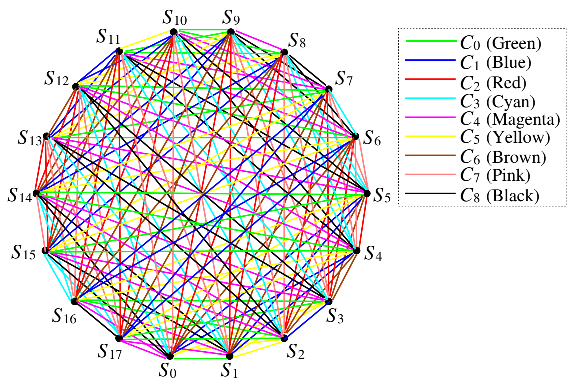


(1) (2) (3) (4)

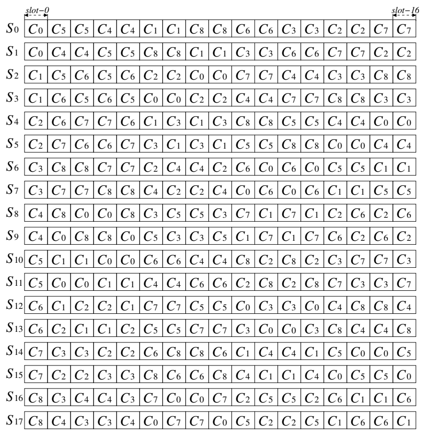
Fig. 21. The four different perfect rainbow matchings decomposed from the 4-factor of K_{18} shown in Figure 17 (d). These four PRMs tell how the 18 CH sequences rendezvous with each other in slot-13 to slot-16 respectively by fully utilizing all the 9 rendezvous channels in each hopping slot.



(a)



(b)



	slot−0																slot−16
S_0	C_0	C_5	C_5	C_4	C_4	C_1	C_1	C_8	C_8	C_6	C_6	C_3	C_3	C_2	C_2	C_7	C_7
S_1	C_0	C_4	C_4	C_5	C_5	C_8	C_8	C_1	C_1	C_3	C_3	C_6	C_6	C_7	C_7	C_2	C_2
S_2	C_1	C_5	C_6	C_5	C_6	C_2	C_2	C_0	C_0	C_7	C_7	C_4	C_4	C_3	C_3	C_8	C_8
S_3	C_1	C_6	C_5	C_6	C_5	C_0	C_0	C_2	C_2	C_4	C_4	C_7	C_7	C_8	C_8	C_3	C_3
S_4	C_2	C_6	C_7	C_7	C_6	C_1	C_3	C_1	C_3	C_8	C_8	C_5	C_5	C_4	C_4	C_0	C_0
S_5	C_2	C_7	C_6	C_6	C_7	C_3	C_1	C_3	C_1	C_5	C_5	C_8	C_8	C_0	C_0	C_4	C_4
S_6	C_3	C_8	C_8	C_7	C_7	C_2	C_4	C_4	C_2	C_6	C_0	C_6	C_0	C_5	C_5	C_1	C_1
S_7	C_3	C_7	C_7	C_8	C_8	C_4	C_2	C_2	C_4	C_0	C_6	C_0	C_6	C_1	C_1	C_5	C_5
S_8	C_4	C_8	C_0	C_0	C_8	C_3	C_5	C_5	C_3	C_7	C_1	C_7	C_1	C_2	C_6	C_2	C_6
S_9	C_4	C_0	C_8	C_8	C_0	C_5	C_3	C_3	C_5	C_1	C_7	C_1	C_7	C_6	C_2	C_6	C_2
S_{10}	C_5	C_1	C_1	C_0	C_0	C_6	C_6	C_4	C_4	C_8	C_2	C_8	C_2	C_3	C_7	C_7	C_3
S_{11}	C_5	C_0	C_0	C_1	C_1	C_4	C_4	C_6	C_6	C_2	C_8	C_2	C_8	C_7	C_3	C_3	C_7
S_{12}	C_6	C_1	C_2	C_2	C_1	C_7	C_7	C_5	C_5	C_0	C_3	C_3	C_0	C_4	C_8	C_8	C_4
S_{13}	C_6	C_2	C_1	C_1	C_2	C_5	C_5	C_7	C_7	C_3	C_0	C_0	C_3	C_8	C_4	C_4	C_8
S_{14}	C_7	C_3	C_3	C_2	C_2	C_6	C_8	C_8	C_6	C_1	C_4	C_4	C_1	C_5	C_0	C_0	C_5
S_{15}	C_7	C_2	C_2	C_3	C_3	C_8	C_6	C_6	C_8	C_4	C_1	C_1	C_4	C_0	C_5	C_5	C_0
S_{16}	C_8	C_3	C_4	C_4	C_3	C_7	C_0	C_0	C_7	C_2	C_5	C_5	C_2	C_6	C_1	C_1	C_6
S_{17}	C_8	C_4	C_3	C_3	C_4	C_0	C_7	C_7	C_0	C_5	C_2	C_2	C_5	C_1	C_6	C_6	C_1

(c)

Fig. 22. The final 18-vertex complete graph K_{18} and the final 18 CH-sequences. The 16 PRMs shown in Figure 18 to Figure 21 and the first PRM (shown in (a)) together form the final 18-vertex complete graph K_{18} (shown in (b)). The final 18 CH sequences (shown in (c)) is constructed based on the 17 PRMs, each of which tells how the 18 CH sequences rendezvous with each other in a different hopping slot by fully utilizing all the 9 rendezvous channels.