Supporting Fast Rendezvous Guarantee by Randomized Quorum and Latin Square for Cognitive Radio Networks

Chih-Min Chao, Member, IEEE, and Hsiang-Yuan Fu

Abstract—Cognitive radio network (CRN) has been widely studied because it significantly enhances spectrum access efficiency. An essence issue for CRN communications is to provide rendezvous between two nodes. An easy but impractical way to achieve rendezvous is to use a dedicated channel to exchange control messages. A better way to provide rendezvous is to exploit channel hopping. Most existing channel hopping solutions suffer from poor system performance. In this paper, we propose a novel distributed channel hopping protocol, Randomized Quorum and Latin square Channel Hopping (RQL). Utilizing the concepts of quorum systems, latin squares, and a pseudo random number generator linear congruential generator (LCG), RQL efficiently provides rendezvous guarantee and balanced channel utilization. The concept of quorum systems is utilized to guarantee balanced rendezvous among nodes while the concept of latin squares and LCG is used to share the rendezvous among channels and to increase channel utilization, respectively. RQL is considered a flexible and robust solution that provides rendezvous guarantee for any pair of nodes in a CRN. Analytical and simulation results verify that RQL performs better in terms of time to rendezvous and network throughput when comparing to existing rendezvous protocols, L-QCH, ACH, and QLCH.

Index Terms—Cognitive Radio Networks, Channel Hopping, Rendezvous Guarantee, Quorum System, Latin Squares, Linear Congruential Generator

I. INTRODUCTION

The rapid development of wireless applications makes wireless spectrum a precious resource. However, a large portion of licensed spectrum is underutilized. Cognitive radio has received a lot of attention recently since it enables more efficient spectrum allocation and utilization. In cognitive radio networks (CRNs), an unlicensed user (secondary user, SU\(^1\)), is allowed to access the licensed spectrum which is not being used by any licensed user (primary user, PU). Exploiting cognitive radio technology, SUs are capable of recognizing spectrum holes and are able to hop between them without causing operation interruption of PUs. That is, SUs are capable of detecting the existence of PUs through spectrum sensing techniques [1] and adapting their radio parameters to exploit spectrum opportunities without interfering with PUs [2]. The sensing task is executed across a wide range of spectrum to identify the spectrum holes. SUs are capable of opportunistic utilization of the spectrum hole. Due to the dynamic nature of PUs’ activities, spectrum sensing is usually performed periodically to keep the occupancy information up-to-date. Cognitive radios (CRs) have been proposed as a possible solution to improve spectrum utilization via opportunistic spectrum sharing. In the literature, two cognitive spectrum access models have received a lot of attention: spectrum overlay and spectrum underlay. In an overlay radio cognitive radio system, SUs opportunistically utilize licensed spectrum holes without interfering with any PU. In an underlay cognitive radio system, an SU can coexist with an active PU in a licensed band if the interference caused to the PU is under a given threshold [3]. In this paper, we consider an overlay cognitive radio system. That is, when PUs appear, the SUs must immediately vacate the spectra being used.

In a CRN where multiple channels can be used by each SU, two SUs have a rendezvous if they tune to the same channel at the same time. Providing a rendezvous between any pair of nodes is a premise for their data communications. However, because SUs may tune to different channels, providing rendezvous between any pair of SUs in a CRN is nontrivial. There exist many multi-channel protocols that provide rendezvous guarantee between any pair of users in traditional wireless networks [4]–[6]; however, these solutions cannot be applied to SUs in CRNs because they do not consider the PU occupancy issue. Existing solutions to the rendezvous problem for SUs in CRNs can be classified by the following two factors:

- **with or without common control channel (CCC)**: Whether nodes use a dedicated CCC or not.
- **with or without rendezvous guarantee**: Whether a rendezvous between any pair of nodes is guaranteed or not.

Based on this classification, existing rendezvous solutions (and ours) can be categorized in Table I.

Note that some rendezvous protocols enable a node to have a rendezvous to parts (instead of all) of the other nodes within a limited time interval. For example, in OS-MAC [7], nodes in the same SU group (SUG) can communicate with each other.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>CLASSIFICATION OF CRN RENDEZVOUS PROTOCOLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/ guarantee</td>
<td>w/o guarantee</td>
</tr>
<tr>
<td>w/ CCC</td>
<td>[7], [8]</td>
</tr>
<tr>
<td>w/o CCC</td>
<td>[12]–[14]</td>
</tr>
</tbody>
</table>

Copyright (c) 2015 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.

\(^1\)The terms SU and node may be used interchangeably to represent a secondary user in this paper.
When a member in an SUG transmits, only the nodes in the same group receive it. In other words, a node cannot have a rendezvous with nodes in different SUGs. In AMRCC [12], each node has a randomized channel hopping sequence and is not guaranteed to have a rendezvous to any other node.

Some protocols [7]–[11] use one or a few dedicated channels as the common control channel(s) to exchange control information. For example, MMAC-CR [8], a revision of MMAC, is an energy-efficient multichannel MAC protocol for CR networks. Nodes running MMAC-CR use a common control channel to share their channel sensing results, in addition to control information exchanging. Using spectral opportunities in licensed channels, MMAC-CR improves network performance. Utilizing a CCC is an easy way to solve the rendezvous problem but a major concern of such protocols is that a CCC may not exist since different users have different available channels. Besides, even if a CCC exists, the single control channel usually becomes a bottleneck and may suffer from security attacks (such as jamming attacks). There are quite many solutions provide rendezvous without using a CCC. Some of them do not guarantee a rendezvous between any pair of nodes [12]–[14] while some others do [15]–[26]. In general, these schemes utilize some sort of channel hopping mechanism where each node switch channels according to its predefined channel sequence. In channel hopping protocols, time is divided into a series of slots. Nodes switch to a particular channel and stay tuned during a slot. Two nodes have a rendezvous when they switch to the same available channel at the same time slot. In these rendezvous-guaranteed protocols, some only provide partial rendezvous guarantee [15]–[22] in that a rendezvous occurs only on some channels. Still other protocols [23]–[26] provide complete rendezvous guarantee in that any pair of nodes can always rendezvous on all the channels. For example, Quorum-based channel hopping (QCH) [23] is a well-known quorum-based channel hopping protocol. A rendezvous between any pair of nodes is guaranteed because of the intersection property of quorum systems. L-QCH is one of synchronous variation for QCH tries to minimize the channel load. Asynchronous channel hopping (ACH) [24] operates similar to a grid quorum system. Each node running ACH creates two grids, one is used when the node acts a receiver while the other is used when it acts as a sender. The intersection between a column and a span in a grid where a span consists of one element from each column is utilized. A node’s channel hopping sequence is obtained from the grid being used. The quorum and latin square channel hopping (QLCH) scheme [25] also utilizes the concepts of quorum systems and latin squares to provide complete rendezvous guarantee. A quorum system is utilized to guarantee balanced rendezvous among nodes while a latin square is used to share the rendezvous among different channels. The complete rendezvous guarantee solutions are considered more flexible and robust because they provide higher communication probabilities when some channels are occupied by PUs. Existing complete rendezvous guarantee protocols suffer from either high time to rendezvous (TTR) or low channel utilization, which reduces system performance.

Another way to classify rendezvous solutions is whether nodes required to be synchronized or not. Asynchronous protocols are suitable for neighbor discovery during the network initialization phase where a node has no information about the other nodes. Asynchronous solutions usually have higher TTR. On the other hand, the synchronous solutions usually have lower TTR at the expense of synchronization overhead.

In this paper, we propose an efficient channel hopping protocol which is called randomized quorum and latin square channel hopping protocol (RQL) to solve the rendezvous problem in CRNs. Existing channel hopping protocols for CRNs focus on designing a fixed channel hopping sequence for each node. Nodes running RQL may change their original channel hopping sequences dynamically according to their intended receivers. Such flexibility enables nodes running RQL to have improved performance when compared with similar conventional protocols of CRNs. RQL provides complete rendezvous guarantee with low TTR, high channel utilization, and high system throughput. An attractive feature of RQL is that nodes do not need to exchange their channel hopping sequences, which reduces energy consumption. To generate a node’s channel hopping sequence, RQL combines the concepts of quorum systems, latin squares, and pseudo random number generators. The quorum system is utilized to provide uniform rendezvous guarantee among different nodes. The latin square is used to spread such rendezvous over all channels. A pseudo number generator which is called linear congruential generator (LCG) is used to provide high channel utilization.

Each node’s channel hopping sequence is determined by its node identification (ID). A node can calculate another node’s channel hopping sequence if its ID and the time slot offset (time slot difference between two nodes) are available. To obtain this information, RQL can cooperate with an existing asynchronous rendezvous-guaranteed channel hopping protocol, such as Jump-stay [19] or ACH [24], at the initialization phase. That is, a node can apply Jump-stay or ACH at its initialization phase. After that, since RQL performs better, a node can apply RQL to enjoy improved performance.

The main contributions of this paper are summarized as follows.

1) Without using any CCC, an asynchronous CRN rendezvous protocol providing more rendezvous among SUs and increased channel allocation is proposed (see Section III).
2) Prove that RQL provides complete rendezvous guarantee (see Theorem 3 in Section III-C).
3) Provide theoretical analysis and simulation to verify the superiority of RQL in TTR and throughput (see Section IV and Section V).

II. Preliminary

We introduce the main concepts including quorum systems, block design, latin squares, and linear congruential generator

2The proposed scheme works on top of several conventional techniques in CR, such as spectrum sensing and spectrum sharing. Specifically, when two SUs have a rendezvous, spectrum sensing is applied to decide whether the licensed spectrum is available or not. If so, they share the common available channel through a single channel MAC protocol (such as IEEE 802.11 DCF).
3A neighbor’s ID and time slot offset can be obtained in the network initialization phase or when a neighbor discovery scheme is executed periodically.
utilized in our channel hopping mechanism in this section.

A. Quorum Systems

Quorum systems [27] have been widely used to provide a uniform and reliable coordination among users in distributed systems [28] and MAC protocol design in wireless networks [5], [29], [30]. A quorum system can be defined as follows.

**Definition 1.** Given an universal set \( U = \{0, \ldots, y - 1\} \), a quorum system \( Q \) under \( U \) is a collection of non-empty subsets of \( U \), each is called a quorum, which satisfies the intersection property:

\[
\forall G, H \in Q : G \cap H \neq \emptyset.
\]

For example, \( Q = \{\{0,1\},\{1,2\},\{0,2\}\} \) is a quorum system under \( U = \{0, 1, 2\} \). There are many quorum systems, such as grid quorum systems, torus quorum systems, and cyclic quorum systems. There are many kinds of quorums, such as the majority-based quorum, the tree-based quorum, the grid-based quorum, and others. In this paper, cyclic quorum systems are used to design our protocol since it can provide an equal opportunity for nodes to transmit and to receive packets.

A cyclic quorum system can be constructed from a difference set. A difference set and a cyclic quorum system can be defined as follows.

**Definition 2.** A subset \( D = \{d_1, \ldots, d_k\} \) of \( Z_y \) is called a difference set under \( Z_y \) if for every \( e \neq 0 \pmod{y} \) there exist at least two different elements \( d_i \) and \( d_j \in D \) such that \( d_i - d_j \equiv e \pmod{y} \).

For example, \( D = \{0,1,3\} \) under \( Z_7 \) is a difference set since \( 1 \equiv 1 - 0, 2 \equiv 3 - 1, 3 \equiv 3 - 0, 4 \equiv 0 - 3 \pmod{7}, 5 \equiv 1 - 3 \pmod{7}, \) and \( 6 \equiv 0 - 1 \pmod{7} \).

**Definition 3.** Given any difference set \( D = \{d_1, \ldots, d_k\} \) under \( Z_y \), the cyclic quorum system defined by \( D \) is \( Q = (G_0, \ldots, G_{y-1}) \), where \( G_i = \{d_1 + i, \ldots, d_k + i\} \pmod{y}, i = 0, \ldots, y - 1 \).

For instance, a difference set \( D = \{0,1,3\} \) under \( Z_7 \), the cyclic quorum system defined by \( D \) is \( Q = \{G_0, G_1, \ldots, G_6\} \), where \( G_0 = D, G_1 = \{1,2,4\}, G_2 = \{2,3,5\}, G_3 = \{3,4,6\}, G_4 = \{0,4,5\}, G_5 = \{1,5,6\}, G_6 = \{0,2,6\} \).

B. Block Design

Block design (or combinatorial design) is a theory of selecting elements from a finite set into subsets to satisfy certain properties [31]. Block design has been widely used in wireless networks, such as scalable key distribution [32], MAC protocol design [33], and channel assignment [34]. A balanced incomplete block designs (BIBD) is defined as follows.

**Definition 4.** A BIBD is an arrangement of \( v \) distinct objects into \( b \) blocks such that

1) each block contains exactly \( k \) distinct objects,
2) each object occurs in exactly \( r \) different blocks, and
3) every pair of distinct objects \( i \) and \( j \) occurs together in exactly \( \lambda \) blocks.

A BIBD with parameters \( v, b, r, k, \lambda \) can be represented by a 5-tuple \((v, b, r, k, \lambda)\).

A BIBD \((v, b, r, k, \lambda)\) satisfies \( bk = vr \) and \( r(k-1) = \lambda(v-1) \). A BIBD with \( v = b \) and \( r = k \) is called a symmetric design and can be represented by \((v, k, \lambda)\). A symmetric BIBD can be used to create a group of subsets from \( v \) distinct elements such that each subset contains exactly \( k \) distinct elements while two different subsets contain exactly \( \lambda \) common elements. There exist many symmetric designs, such as the affine plane \((q^2, q, 1)\), the projective plane \((q^2 + q + 1, q + 1, 1)\), and the Hadamard design \((4q + 3, 2q + 1, q)\). Block design can be used to find out the difference set being utilized in our protocol. For example, a projective plane \((7, 3, 1)\) (with \( q = 2 \)) which is also a Hadamard design (with \( q = 1 \)), is a symmetric BIBD and the set \( \{1,2,4\} \) is a difference set under \((7, 3, 1)\). In this paper, we use \( \Theta(v, k, \lambda) \) to represent a cyclic quorum system generated by BIBD \((v, k, \lambda)\).

C. Latin Squares

Latin squares have been extensively used in algebra, statistics, finite geometrics, coding theory, and combinatorial design theory [35]. Latin squares have also been applied for designing wireless MAC protocols [36] and network coding schemes [37]. The definition of a latin square is listed below.

**Definition 5.** A latin square of order \( d \) is an \( d \times d \) matrix composed by \( d \) different symbols such that each symbol occurs exactly once in each row and column.

For example, the squares \( A \) and \( B \) shown below are two latin squares of order 4 with symbols 1, 2, 3, and 4.

\[
A = \begin{bmatrix}
1 & 2 & 3 & 4 \\
4 & 1 & 2 & 3 \\
3 & 4 & 1 & 2 \\
2 & 3 & 4 & 1 \\
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
4 & 3 & 1 & 2 \\
2 & 1 & 3 & 4 \\
1 & 2 & 4 & 3 \\
3 & 4 & 2 & 1 \\
\end{bmatrix}
\]

If we permute the rows, the columns, or the symbols of a latin square, we obtain a new latin square. A method based on the modulo multiplication group theory is introduced to generate latin squares [36]. Given two vectors \( A \) and \( B \) of order \( d \), in which both vectors are permutation arrays of the symbol set \( \{0, 1, \ldots, d - 1\} \), we can generate a latin square of order \( d \), \( L = A^T \cdot B \mod d \), where \( d \) is a prime number.

D. Linear Congruential Generator

The LCG is a method that utilizes linear equation to yield a sequence of randomized numbers [38]. LCG is one of the well-known pseudo-random number generator algorithms which generate a sequence of numbers approximating the properties of random numbers. A LCG is defined by the recurrence relation as \( X_{i+1} = (aX_i + c) \mod m \), where \( (X_0, X_1, \ldots) \) is a sequence of pseudo-random values, the ‘modulus’ \( m \), the ‘factor’ \( a \), and the ‘increment’ \( c \) are given constant integers. \( X_0 \) is also called the initial seed of the LCG. The LCG
generated by parameters $a$, $c$, $m$, and $X_0$, which can be represented by a 4-tuple $(a, c, m, X_0)$, satisfies the constraints:

$0 < a < m$, $0 < c < m$, and $0 < X_0 < m$.

For the LCG $(a, c, m, X_0)$, the maximum period is determined by parameters $a$, $c$, and $m$. The maximum period is $m$ and the period can be much less than $m$ for some values of $a$. Here we introduce a well-known Hull-Dobell Theorem which finds the values of these parameters such that the produced LCG has the maximum period [39].

**Theorem 1. (Hull-Dobell Theorem)** A complete period of the LCG $(a, c, m, X_0)$ with mixed congruential method contains all residues mod $m$, if and only if

1. $c$ and $m$ are relatively prime, i.e. $\gcd(c, m) = 1$;
2. $a \equiv 1 \pmod{p}$ for all prime factors $p$ of $m$, i.e. $p \mid (a-1)$ for every prime $p$ such that $p \mid m$;
3. $a \equiv 1 \pmod{4}$ if 4 is a factor of $m$, i.e. if $4 \mid m$ then $4 \mid (a-1)$.

In this paper, we call a LCG satisfying the Hull-Dobell theorem an HD-LCG.

We also need to define a cyclic LCG system as follows.

**Definition 6.** Given a sequence $A = (a_0, a_1, \ldots, a_i)$, let $\text{rotate}(A, j)$ be a function to rotate $A$ right $j$ times. That is, $\text{rotate}(A, j) = (a_{i-j+1}, a_{i-j+2}, \ldots, a_i, a_0, \ldots, a_{i-j})$. We say $B$ has a rotation relation with $A$, denoted as $A \circ B$, if $B = \text{rotate}(A, j)$, $j \in \mathbb{N}$.

**Definition 7.** Given an HD-LCG $(a, c, m, x)$ (denoted as $\text{lcg}(x)$), the set $\Omega(a,c,m) = \{\text{lcg}(0), \text{lcg}(1), \ldots, \text{lcg}(m-1)\}$ is a cyclic LCG system if any two LCGs of the cyclic LCG system have a rotation relation. That is, $\forall \text{lcg}(i), \text{lcg}(j) \in \Omega(a,c,m), \text{lcg}(i) \circ \text{lcg}(j)$

Note that a cyclic LCG system satisfies the rotation closure property. That is, for a cyclic LCG system $\Omega$, $\forall A \in \Omega$ and $j \in \mathbb{N}$, $\text{rotate}(A, j) \in \Omega$.

### III. THE PROPOSED APPROACH

Without loss of generality, we consider a CRN with $n$ rendezvous channels, labeled from 0 to $n-1$. Nodes are randomly scattered in the CRN. Each node has a unique ID (the ID of node $u$ is $u$). Global synchronization among nodes is not necessary. Each node knows all of its neighbors’ IDs and time slot offsets. Each node is equipped with a single half-duplex transceiver which can be switched to any channel dynamically. The IEEE 802.11 DCF is adopted as the MAC scheme when two nodes communicate. Some important notations being used in this paper are listed in Table II.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Omega_{(v,k,\lambda)}$</td>
<td>A cyclic quorum system generated by BIBD $(v, k, \lambda)$.</td>
</tr>
<tr>
<td>$\Omega_{(a,c,m)}$</td>
<td>The cyclic LCG system formed by HD-LCG $(a, c, m, X_0)$.</td>
</tr>
<tr>
<td>$\mathcal{S}_u(s)$</td>
<td>The slot state of node $u$ at slot $s$.</td>
</tr>
<tr>
<td>$L_u$</td>
<td>The set of node $u$'s intended receivers.</td>
</tr>
<tr>
<td>$I^f_u(s)$</td>
<td>Node $u$’s initial channel for slot $s$ in frame $f$.</td>
</tr>
<tr>
<td>$H_u(s)$</td>
<td>Hopping channel of node $u$ at slot $s$.</td>
</tr>
<tr>
<td>$\mathcal{F}_u^f(f)$</td>
<td>The pattern being used by node $u$ in frame $f$ of superframe $F$.</td>
</tr>
<tr>
<td>$\mathcal{R}_{u,w}^f(f)$</td>
<td>The rendezvous slot set of nodes $u$ and $w$ in frame $f$ of superframe $F$.</td>
</tr>
<tr>
<td>$D_{u,w}^f$</td>
<td>The set of frames in which nodes $u$ and $w$ use different patterns in superframe $F$.</td>
</tr>
<tr>
<td>$T_{mf}$</td>
<td>The interval of the longest continuous missing frames.</td>
</tr>
<tr>
<td>$T_{ms}$</td>
<td>The interval of the longest continuous missing slots in a rendezvous frame.</td>
</tr>
<tr>
<td>$\Gamma(h)$</td>
<td>The length of LRS when using Hadamard design $(h, k, \lambda)$.</td>
</tr>
</tbody>
</table>

#### A. RQL Frame Structure and Pattern Allocation

In the proposed RQL scheme, time is divided into a series of superframes each of which consists of $m$ frames, labeled from 0 to $m-1$. Each frame consists of $n$ slots, labeled from 0 to $n-1$. The length of a time slot is set to equal to that of a sensing period. That is, the status of the channel (idle or occupied) remains unchanged during a time slot. Because global synchronization among nodes is not necessary, the frames for different nodes may not be aligned. Nodes running RQL adopt this frame structure. That is, different SUs have the same RQL frame architecture. Each node $u$ is assigned an initial channel hopping sequence which determines the channel node $u$ should be switched to at a particular slot. The initial channel hopping sequence allocation mechanism will be described in Section III-B. Based on the initial channel hopping sequence, the channel assigned to a node at a particular slot is called the initial channel of the slot. A node may also switch to a channel other than the initial channel. The channel a node actually switches to is called the node’s hopping channel of the slot. A node also partitions time slots into default slots and switching slots. At default slots, the hopping channels are the initial channels. At switching slots, a node can switch to any of its intended receivers’ initial channel to start a transmission if the corresponding receiver is tuned to its initial channel and the channel is available. If the selected hopping channel is unavailable or a node has no pending packets at a switching slot, the hopping channel remains unchanged. That is, there is no channel hopping when entering the current switching slot.

To facilitate our presentation, we define a Boolean valuable, $S_u(s)$, to represent the slot state of node $u$ at slot $s$, $S_u(s)$ is equal to 1 if $s$ is a default slot for node $u$ and 0 otherwise. Also, we define $L_u$ to be the set of node $u$’s intended receivers and $I^f_u(s)$ to be node $u$’s initial channel of slot $s$ of frame $f$. The algorithm of determining a node $u$’s hopping channel at slot $s$ (denoted as $H_u(s)$) is shown in Algorithm 1.

Each node executes algorithm 1 at every slot. For each slot, if the slot is a default slot, node $u$ hops to its initial channel (lines 1 to 2). Otherwise, the hopping channel remains unchanged (line 4). If $u$ has data to send, it will pick a node with available initial channel from its intended receivers as the receiver (lines 5 to 15). If a receiver is found, node $u$ will hop to the receiver’s initial channel (lines 10 to 12). In general, a node is waiting for transmission requests at its default slots and sends transmission requests at its switching slots.

RQL utilizes a cyclic quorum system under $Z_n$ to assign
Algorithm 1 Hopping Channel Determination

Input: slot s, node u, node u’s intended receiver set L_u;
Output: H_u(s);
1: if S_u(s) = 1 then
2: H_u(s) ← I_u^s(s);
3: else
4: H_u(s) ← H_u(s - 1);
5: if L_u ≠ ∅ then
6: Q = L_u;
7: repeat
8: randomly pick a node w ∈ Q;
9: Q = Q \ w;
10: if S_u(w) = 1 then
11: H_u(s) ← I_u^w(s);
12: return H_u(s);
13: end if
14: until Q = ∅;
15: end if
16: end if
17: return H_u(s);

default and switching slots. A node running RQL selects a quorum G_i for each frame and the corresponding slots of G_i are assigned as the default slots of the node. To provide an equal opportunity for a node to communicate to different nodes, RQL uses a cyclic quorum system generated by a symmetric BIBD to assign each node’s default slots. The selection mechanism of G_i will be described in Section III-B because it is affected by the number of channels. The selected quorum is also called a default slot pattern, we call it a pattern for short hereafter.

To provide rendezvous guarantee and high channel utilization, the patterns used for different frames are different. RQL uses an HD-LCG to generate each node’s pattern hopping sequence. Specifically, for each node, an initial random number sequence of node u is generated first which is then mapped to a sequence of patterns used for different frames are different. RQL uses an HD-LCG to generate each node’s pattern hopping sequence. Specifically, for each node, an initial random number sequence of node u is generated first which is then mapped to a sequence of patterns being used in consecutive frames. Let X_u(i) represent the i-th element of node u’s initial random number sequence, the initial random number sequence of node u is generated by a LCG (a, c, m, u) and

\[ X_u(f) = \begin{cases} u \mod m & \text{if } f = 0, \\ (ax_u(f - 1) + c) \mod m & \text{otherwise}. \end{cases} \]  

Refer to Theorem 1, the values of a, c and m follow the Hull-Dobell theorem in order to get the maximum period of pseudo-random number sequence. Given two nodes u, w < m, the initial random number sequences of u and w are totally different, that is, no slot two initial random number sequences have the same number. In a particular frame, the pattern being used by node u is determined by u’s initial random number sequence. Specifically, the pattern being used by node u in frame f of superframe F, denoted as P_u^f(f), is given by

\[ P_u^f(f) = G_i, \text{ where } i = (X_u(f) + F) \mod n. \]  

In this pattern allocation mechanism, if the pattern being used by a node in frame f of superframe F is G_i, the pattern being used by the node in frame f of superframe F + 1 is G_j, where j = (i + 1) mod n. Such a design achieves even pattern allocation. Fig. 1 is an example of pattern hopping in superframe 0 for a CRN with seven channels, using a cyclic quorum system Θ_{(7,3,1)} and the cyclic LCG system Ω_{(5,1,8)}.

The first eight initial random number sequence of node 0 and node 2 is \((0, 1, 6, 7, 4, 5, 2, 3)\) and \((2, 3, 0, 1, 6, 7, 4, 5)\), respectively. The complete pattern hopping sequence for node 0 and node 2 is thus \((G_0, G_1, G_6, G_0, G_4, G_5, G_2, G_3)\) and \((G_2, G_3, G_0, G_1, G_6, G_0, G_4, G_5)\), respectively. Note that the pattern hopping sequences of nodes 0 and 8 are exactly the same since they have the same initial random number sequence in a LCG system with m = 8. The number in a slot is the initial channel of the slot (the channel allocation scheme will be described in Section III-B).

Fig. 2 is an example of the RQL operation where nodes 0 and 1 have pending packets to each other. The number at the upper left corner of a switching slot is the initial channel of the slot while the number at the center is the hopping channel of the slot. In frame 0, nodes 0 and 1 select G_0 and G_1 as their patterns, respectively. For the default slots of node 0 (slots 0, 1, and 3) and node 1 (slots 1, 2, and 4), both nodes’ hopping channels are set to their initial channels. The switching slots for node 0 in frame 1 are slots 2 and 4 and the hopping channel of node 0 for slot 2 and slot 4 is channel 3 and channel 5, respectively. This enables node 0 to transmit to node 1 at slots 2 and 4. Similarly, the hopping channel of node 1 for slot 0 and slot 3 is channel 0 and 3, respectively. Note that the hopping channels for both nodes 0 and 1 remain unchanged at time slots 5 and 6 if both nodes do not have pending packets to the other. Similar channel hopping can be found in frame 1.

B. RQL Channel Allocation

To allocate each node’s hopping channels, RQL uses the concept of latin squares to produce balanced channel utilization. To allocate n channels, a latin square of \(n \times n\) with elements \(e_{i,j} = (i + j) \mod n, i, j = 0, \ldots, n-1\) is utilized, as shown in Fig. 1. The columns of the latin square represent time slots in a frame. The symbols in the row represent a node’s initial channel hopping sequence. The initial channel of node u for slot s of frame f, I_u^f(s), is given by

\[ I_u^f(s) = (r_u + s) \mod n. \]  

\(^4\)Channel 3 and channel 5 is the corresponding initial channel of node 1’s default slots 2 and 4, respectively.
where \( r_u = u \mod m \). For example, in a CRN with \( n = 7 \) and \( G_0 = \{0, 1, 3\} \), as shown in Fig. 1, the default slots for node 1 of frame 0 are slots 1, 2, and 4 while the initial channel hopping sequence for slots 0 to 6 of frame 0 is 1, 2, 3, 4, 5, 6, and 0, respectively.

Two nodes \( u \) and \( w \) are defined to be equivalent if \( u \equiv w \pmod{m} \), that is \( r_u = r_w \). Note that equivalent nodes will have the same initial random number sequence and thus the same pattern hopping sequence. Also, they have the same initial channel hopping sequence. That is, equivalent nodes have a rendezvous with each other at any default slot.

As described in Section III-A, RQL utilizes a quorum system generated by a symmetric BIBD to determine a node’s default/switching slots. Note that a symmetric BIBD \((v, k, \lambda)\) can be found only for some specific values of \( v \). Take Hadamard design which is used in RQL as an example, the design can be applied when \( v = 4q + 3 \) where \( q \) is a positive integer. For \( q = 1 \) and 2, we can find the corresponding Hadamard design \((7, 3, 1)\) and \((11, 5, 2)\), respectively. A difference set that satisfies a Hadamard design is called a Hadamard difference set. For example, \( \{1, 2, 4\} \) is a Hadamard difference set satisfying the Hadamard design \((7, 3, 1)\). No Hadamard design can be found when \( v \) is equal to 8, 9, or 10.

To apply RQL to any number of channels, RQL uses the least difference set satisfying the Hadamard design \((7, 3, 1)\). Specifically, \( P_u^w(f) \) is modified by

\[
P_u^w(f) = G_i, \quad \text{where} \quad i = (X_u(f) + F) \mod h,
\]

And \( I_u^w(s) \) is modified by

\[
I_u^w(s) = (r_u + s + hf) \mod n.
\]

To summarize, RQL uses an \( m \times h \) matrix to determine the pattern hopping sequences and an \( n \times n \) Latin square to determine the channel hopping sequences. Note that the \( n \times n \) Latin square is fixed, while \( m \times h \) matrix is varied frame by frame. Fig. 3 is an example of pattern hopping as well as channel hopping of the first two superframes in a CRN with six channels, using the cyclic LCG system \( \Omega_{(5,1,8)} \) and a cyclic quorum system \( \Theta_{(2,1,4)} \). The \( 8 \times 7 \) matrices, drawn by double solid lines, determine the patterns while the \( 6 \times 6 \) grids, drawn by single solid lines, determine the channel hopping sequences.

In RQL, Hadamard design \((h, k, \lambda)\) is adopted because of its outstanding performance (performance analysis will be presented in Section IV). The Hadamard design with \( n \) values from 2 to 106 is listed in Table III. Note that for a Hadamard design \((h, k, \lambda)\), \( k = (h - 1)/2 \) and \( \lambda = (h - 3)/4 \). Note that we do not find Hadamard design for some \( h \) values, such as \((27, 13, 6)\) and \((39, 19, 9)\). On the other hand, to provide rendezvous guarantee, the cyclic LCG system \( \Omega_{(a,c,m)} \) used in RQL is selected such that the following two constraints are satisfied: (1) \( h \) and \( m \) are relatively prime and (2) \( c < a < h < m < ah \).

C. RQL Properties

Nodes running RQL use a cyclic quorum system (defined by a difference set) to determine default/switching slots in a frame. There is an equal number of overlapping slots between any pair of nodes using different quorums. According to the definition of the symmetric BIBD, there are \( 2(k-\lambda) \) different elements between any two different quorums in a cyclic quorum system generated by a symmetric BIBD \((v, k, \lambda)\). That is, two nodes using different patterns generated by a cyclic quorum system \( \Theta_{(v,k,\lambda)} \) have \( 2(k-\lambda) \) overlapping slots. Next, we show that two nodes running RQL are guaranteed to have a rendezvous.

**Theorem 2.** RQL is a rendezvous-guaranteed protocol.

**Proof.** Depending on whether any two nodes \( u \) and \( w \) are equivalent or not, the proof is divided into two parts:

1. **Nodes \( u \) and \( w \) are equivalent** (\( u \equiv w \pmod{m} \)).
   - We have \( P_u^w(f) = P_w^u(f) \) and \( I_u^w(s) = I_w^u(s) \), \( \forall F, f, s \).
   - This means that \( u \) and \( w \) have the same initial channels for all the slots and thus have a rendezvous at every default slot.

   **ii.** **Nodes \( u \) and \( w \) are not equivalent.**
   - Here we first prove that nodes \( u \) and \( w \) have a rendezvous if \( P_u^w(f) \neq P_w^u(f) \). Suppose that RQL uses a cyclic quorum system \( Q = \{G_0, \ldots, G_{h-1}\} \) under \( Z_h \). We need to prove that at least one of \( u \)'s default slots overlaps one of \( w \)'s switching slots \( P_u^w(f) \cap P_w^u(f) \neq \emptyset \), and vice versa. Let \( P_u^w(f) = G_i \) and \( P_w^u(f) = G_j, i \neq j \). Because \( G_i \cap G_j \neq \emptyset \) iff \( G_i \neq G_j \), we have \( P_u^w(f) \cap P_w^u(f) \neq \emptyset \) iff \( P_u^w(f) \neq P_w^u(f) \). Similarly, \( P_u^w(f) \neq P_w^u(f) \) iff \( P_u^w(f) \neq P_w^u(f) \) which proves that \( u \) and \( w \) have a rendezvous when \( P_u^w(f) \neq P_w^u(f) \).

   Next, we prove that nodes that are not equivalent will never have the same pattern hopping sequence. We prove this by contradiction. Without loss of generality, we consider a network running RQL using the cyclic LCG system \( \Omega_{(a,c,m)} \) and a cyclic quorum system \( \Theta_{(h,k,\lambda)} \). Assume that the initial random number sequences \( lcg(X_0) = \ldots \).
\((X_0, X_1, \ldots, X_{m-g-1}, X_{m-g}, \ldots, X_{m-1})\) and \(lcg(X_g) = (X_g, X_{g+1}, \ldots, X_{m-1}, X_0, \ldots, X_g)\) produce the same pattern hopping sequence, we have \(X_i \equiv X_{j} \pmod{h}\) for \(i \in [0, m-1]\) and \(j \equiv i + g \pmod{m}\). This is equivalent to

\[ X_i = X_j + h z_i, \]

where \(z_i\) is a non-zero integer. Since \(X_i < m\) and \(X_j < m\), \(\forall i, j\), we have \(-m < h z_i < m\). Besides, since \(m < a h\) in RQL LCG selection, we have

\[-a < z_i < a, \forall i \in [0, m-1].\]

Note that \(X_{i+1} = (aX_i + c) \pmod{m}\) can be rewritten as

\[ X_{i+1} = aX_i + c - my_i, \]

where \(y_i\) is a non-negative integer for \(i \in [0, m-2]\). The variable \(y_i\) must satisfy \(my_i \leq aX_i + c\) because \(0 \leq X_{i+1}\). Because \(X_i < m\) and \(c < a\) in RQL LCG selection, we have \(0 \leq y_i < a\). Substitute Eq. (6) \((X_i = X_j + h z_i\) and \(X_{i+1} = X_{j+1} + h z_{i+1}\)) into Eq. (8), we have

\[ X_{j+1} = aX_j + c + h(a z_i - z_{i+1}) - my_i. \]

Combine Eq. (8) and Eq. (9), we have

\[ h(a z_i - z_{i+1}) = m(y_i - y_j). \]

Recall that \(h\) and \(m\) are relatively prime (a constraint of RQL LCG selection) and \(y_i\) is non-negative for all values of \(i\), \(a z_i - z_{i+1}\) must be a multiple of \(m\) and \(y_i - y_j\) must be a multiple of \(h\) for all values of \(i\). On the other hand, since both \(y_i\) and \(y_j\) are less than \(a\), we have \(-a < y_i - y_j < a\). Given \(a < h\) (also a constraint of RQL LCG selection), \(y_i - y_j\) is a multiple of \(h\) only when \(y_j - y_i = 0\). Back to Eq. (10), this implies that \(a z_i - z_{i+1} = 0\) or equivalently, \(z_{i+1} = a z_i, \forall i \in [0, m-1]\). This contradicts to Eq. (7) and proves this theorem. \(\square\)

Theorem 2 verifies that RQL is a rendezvous-guaranteed protocol. Next, we prove that RQL provides complete rendezvous guarantee. To facilitate our proof, we define the rendezvous slot set of nodes \(u\) and \(w\) in frame \(f\) of superframe \(F\) (denoted as \(\mathcal{R}_{u,w}(f)\)) to be set of rendezvous slots for nodes \(u\) and \(w\) in frame \(f\) of superframe \(F\).

\[ \mathcal{R}_{u,w}(f) = \{ s | S_u(s) = 1 \} \text{ if } u \text{ and } w \text{ are equivalent, otherwise, } \mathcal{R}_{u,w}(f) = \{ s | S_u(s) \neq S_w(s) \}. \]

For example, consider a CRN with six channels, using the cyclic LCG system \(\Theta(5,1,8)\) and cyclic quorum system \(\Theta(7,3,1)\) as shown in Fig. 4, the rendezvous slot set for node 0 and node 1 is

\[ \mathcal{R}_{0,1}(0) = \{0, 2, 3, 4\} \text{ and } \mathcal{R}_{0,1}(1) = \{1, 3, 4, 5\}. \]

\textbf{Theorem 3.} RQL provides complete rendezvous guarantee.

\textbf{Proof.} Given any receiver node \(u\), we prove that \(u\) can have a rendezvous with any sender node \(w\) on all the channels. According to theorem 2, \(u\) and \(w\) are guaranteed to have a rendezvous, that is, \(\forall F, \exists f \in [0, h - 1] \ni \mathcal{R}_{u,w}(f) \neq \emptyset\). Without loss of generality, assume that nodes \(u\) and \(w\) have a rendezvous at slot \(s_0\) on channel \(i\) in frame \(f\) of superframe \(F\), that is, \(s_0 \in \mathcal{R}_{u,w}(f)\). From Eq. (4), we have \(s_0 + 1 \pmod{h} \in \mathcal{R}_{u,w}(f)\). Since the initial channel hopping sequence of the same frame in different superframes is the same, nodes \(u\) and \(w\) have a rendezvous on channel \(i + 1 \pmod{n}\) in frame \(f\) of superframe \(F + 1\). Because \(h \geq n\), a frame consists of all the \(n\) channels and thus nodes \(u\) and \(w\) are guaranteed to have a rendezvous in \(n\) successive superframes. This proves that RQL provides complete rendezvous. \(\square\)

Using a quorum system and a latin square, RQL has several advantages. First, different pairs of channel hopping sequences have scattered overlaps of default/switching slots. This reduces transmission collisions. Second, once the IDs and time slot offsets of its neighbors are known, a node can calculate its intended receiver’s channel hopping sequence without message exchanging. Knowing the receiver’s channel hopping sequence, the sender avoids unnecessary transmission attempts and reduces energy waste. This also decreases traffic collisions and increases system throughput. Third, because in a latin square each symbol occurs exactly once in any row, the rendezvous for the same pair of nodes is distributed among different channels.

In RQL, if a node wants to join the RQL operation, it must first obtain the IDs and time slot offsets of its neighbors at its initialization phase. This can be done by applying an existing asynchronous channel hopping protocol, such as ACH and Jump-stay. Nevertheless, we also propose a simple initialization mechanism: Node \(u\) sends \(Hello\) packets through any available channel \(i\). The size of a \(Hello\) packet is quite small and thus multiple \(Hello\) packets can be sent in a slot. A node \(w\) running RQL is very likely to receive a \(Hello\) packet, if \(w\) also resides in channel \(i\). After receiving a \(Hello\) packet, node \(w\) will respond its ID, slot information (slot number, frame number, and superframe number), and its neighbor information to \(u\). With the information, node \(u\) can also try to send \(Hello\) packets to the neighbors of \(w\) to check if they are also \(u\)’s neighbors. If no response is received in channel \(i\) after a predefined time interval, node \(u\) can switch to another available channel \(j\) to transmit \(Hello\) packets. A node intends to join the RQL operation will transmit \(Hello\) packets in at most \(n\) channels. The joining process of \(u\) is considered complete when node \(u\) has checked all the possible neighbors obtained from the responses of the \(Hello\) packets.

When node \(u\) receives other neighbor’s response containing the neighbor’s slot number of the current slot, it calculates the time slot offset to the neighbor. Specifically, let \(s_u\) and \(s_w\) denote the slot number labeled by node \(u\) and \(w\) for the same slot, respectively. The time slot offset of \(u\) to \(w\), denoted as
\( \Delta s_u^w \), can be calculated by \( \Delta s_u^w = s_w - s_u \). Afterwards, node \( u \) can obtain node \( w \)'s slot number by \( s_w = s_u + \Delta s_u^w \). For example, suppose that node \( u \) receives a response with \( s_w = 9 \) at slot 2 (\( s_u = 2 \)), we have \( \Delta s_u^w = 9 - 2 = 7 \). At a later time slot, say time slot 12 (\( s_u = 12 \)), node \( u \) can calculate the slot number of node \( w \) by \( s_w = 12 + 7 = 19 \).

IV. Performance Analysis

Similar to existing rendezvous protocol [15], [23], the metrics being used in the analysis include expected time to rendezvous (ETTR) and maximum time to rendezvous (MTTR) among all the nodes. The ETTR/MTTR between any pair of nodes is defined as the average/longest time interval for their two consecutive rendezvous.

In this analysis, RQL is running in a CRN with \( n \) available rendezvous channels. A cyclic quorum system \( \Theta_{(h,k,\lambda)} \) and the cyclic LCG system \( \Omega_{(a,c,m)} \) are applied. We first analyze the ETTR of RQL. To facilitate our analysis, we define \( DP_{u,w}^F \) to be the set of frames in which nodes \( u \) and \( w \) use different patterns in superframe \( F \). That is,

\[
DP_{u,w}^F = \{ f \mid P_u^F(f) \neq P_w^F(f), \forall f \in F \}. \tag{11}
\]

According to Theorem 2 and the definition of symmetric BIBD, the number of rendezvous slot between nodes \( u \) and \( w \) in a frame \( f \) of superframe \( F \) is given by

\[
|R_{u,w}(f)| = \begin{cases} 
  k & \text{if } u \equiv v \pmod{m}, \\
  2(k - \lambda) & \text{if } f \in DP_{u,w}^F, \\
  0 & \text{otherwise}.
\end{cases} \tag{12}
\]

Because \( \lambda(h-1) = k(k-1) \) for a symmetric BIBD \( (h, k, \lambda) \), we can replace \( \lambda \) by \( \frac{k(k-1)}{h-1} \) in Equ. (12):

\[
|R_{u,w}(f)| = \frac{2k(h-k)}{h-1}, \quad \text{if } f \in DP_{u,w}^F. \tag{13}
\]

For equivalent nodes, a rendezvous occurs at every default slot. Because there are \( k \) default slots in a frame and a frame consisting of \( h \) slots, the ETTR for equivalent nodes \( u \) and \( w \), denoted as ETTR\(_{u\equiv w} \), is equal to \( \frac{h}{k} \) time slots. Note that the probability of nodes \( u \) and \( w \) being equivalent, \( P[u \equiv w] \), is equal to \( \frac{1}{m} \). For nodes that are not equivalent, since there are a total of \( h \) patterns, the probability of two nodes using different patterns in each frame is \( P[P_u^F(f) \neq P_w^F(f)] = \frac{h-1}{h} \). The ETTR for two nodes \( u \) and \( w \) that are not equivalent, denoted as ETTR\(_{u\neq w} \), is defined as the number of slots in a superframe divided by the number of rendezvous in a superframe. That is,

\[
ETTR_{u\neq w} = \frac{m \cdot h}{\sum_{j=0}^{m-1}|R_{u,w}^F(j)|} = \frac{m \cdot h}{(h-1)m + \frac{2k(h-k)}{h-1}} = \frac{h^2}{2k(h-k)}. \tag{14}
\]

We define the RQL scheme without using a particular BIBD design the normal-formed RQL, denoted as RQL\(_{(NF)} \). The ETTR for any two nodes running RQL\(_{(NF)} \) is thus given by

\[
ETTR_{RQL_{(NF)}} = P[u \equiv w] \cdot ETTR_{u\equiv w} + P[u \neq w] \cdot ETTR_{u\neq w}.
\]

Thus, we have

\[
ETTR_{RQL_{(NF)}} = \frac{1}{m} \cdot k \cdot m - 1 \cdot \frac{h^2}{2k(h-k)} = \frac{(m+1)h^2 - 2kh}{2mk(h-k)}. \tag{15}
\]

Different BIBD designs have different \( (h, k, \lambda) \) values which produce different values of ETTR. To find the minimum ETTR, we have compared three different block designs: affine plane \( (q^2, q, 1) \), projective plane \( (q^2 + q + 1, q + 1, 1) \) and Hadamard design \( (4q + 3, 2q + 1, q) \). We found that Hadamard design has the lowest ETTR values among these three designs. Therefore, the Hadamard design is chosen to construct the quorum system being used in RQL. The ETTR for RQL using Hadamard design, denoted as ETTR\(_{RQL} \) are given by

\[
ETTR_{RQL} = 2 + \frac{2m}{m(h^2 - 1)}. \tag{16}
\]

Note that \( h \geq 7 \) \( \text{(refer to Table III)} \) and \( h < m \), the upper bound of ETTR\(_{RQL} \) is given by

\[
ETTR_{RQL} < 2 + \frac{2m}{m(h^2 - 1)} = 2 + \frac{2}{h^2 - 1} \leq 2.08. \tag{17}
\]

When \( h \) is large, ETTR\(_{RQL} \) converges to 2. To the best of our knowledge, this is the smallest ETTR value for existing rendezvous protocols.

To calculate MTTR between two nodes \( u \) and \( w \), we only need to consider the situation that \( u \) and \( w \) are not equivalent because equivalent nodes have a rendezvous at every frame. Two scenarios need to be considered separately. Firstly, we consider the time interval between nodes \( u \) and \( w \) having no rendezvous because they have identical pattern but different initial channels. We define the frames that nodes \( u \) and \( w \) have identical pattern but different initial channels to be the missing frames for nodes \( u \) and \( w \). The interval of the longest continuous missing frames is denoted as \( T_{mf} \). Secondly, we consider the MTTR results from different patterns. We define the frames that nodes \( u \) and \( w \) have different patterns to be the rendezvous frames and the slots that two nodes have the same slot state to be the missing slots. The longest interval of continuous missing slots between nodes \( u \) and \( w \) in a rendezvous frame is denoted as \( T_{ms} \). The MTTR of RQL happens when the longest continuous missing frames are in the middle of two rendezvous frames. That is, MTTR\(_{RQL} = T_{ms} + T_{mf} + T_{ms}

To find \( T_{mf} \), given a initial random number sequence \( lcg(X_0) = (X_0, X_1, \ldots, X_{m-1}) \) wherein two subsequences of \( lcg(X_0) \), \( (X_0, X_1, \ldots, X_b) \) and \( (X_a, X_{a+1}, \ldots, X_{a+b}) \), generate the same pattern hopping subsequence. That is, \( X_i \equiv X_{a+i} \pmod{h} \), for \( i \in [0, b] \) and \( X_{a+b} \neq X_{a+b+1} \pmod{h} \). Let \( X_1 = X_{a+b} + hz_i \), where \( i \in [0, b] \) and \( z_i \) is a non-zero integer. Based on the proof of Theorem 2, we have \( z_1 = a_0 b, z_2 = a_1 b, \ldots, z_b = a_{b-1} b \), which means \( z_b = a^b b_0 \). Since \( m > \left| X_b - X_{a+b} \right| = hz_b = \left| ha^b z_0 \right| \), we have \( ha^b < m \) due to \( \left| z_0 \right| \geq 1 \), i.e. \( 0 < b > \log_a \frac{m}{h} \). That is, the upper bound of missing frames is \( \left\lfloor \log_a \frac{m}{h} \right\rfloor + 1 \). Thus, \( T_{mf} = \left( \left\lfloor \log_a \frac{m}{h} \right\rfloor + 1 \right) \cdot h \) slots. Because a larger value of \( m \) suffers from a higher MTTR, a small value of \( m \) is preferred.
In RQL LCG selection, we let \( m < ah \) such that \( T_{ms} \) is limited to one frame, that is, \( \max(T_{ms}) = h \).

To find \( T_{ms} \), we need to calculate the interval of continuous missing slots in a rendezvous frame. We found that finding \( T_{ms} \) is actually the longest repeated substring (LRS) problem [40]. Given a Hadamard difference set being used by RQL, the LRS corresponding to this difference set can be found by using any existing LRS algorithm (say, Suffix Trees [40]). We define \( \Gamma(h) \) to be the length of LRS when using a Hadamard difference set satisfying the Hadamard design \((h,k,\lambda)\). For example, for the Hadamard difference set \( \{0, 1, 3\} \) which satisfies the Hadamard design \((7, 3, 1)\), we have \( \Gamma(7) = 2 \). We do not find a general form to obtain the LRS for different Hadamard difference sets\(^5\). Table III lists \( \Gamma(h) \) for different \( n \) and \( h \) (In RQL, Hadamard design \((h,k,\lambda)\) is applied for each \( n \) where \( h > n \)). This table provides us the upper bound of the MTTR in a rendezvous frame (\( \max(T_{ms}) = \Gamma(h) \)). The MTTR for RQL happens when continuous missing frames are in the middle of two rendezvous frames. That is,

\[
\text{MTTR}_{RQL} = T_{mf} + 2T_{ms} = h + 2\Gamma(h) \quad (18)
\]

The ETTR and MTTR comparison for RQL and existing representative solutions, L-QCH, ACH, and QLCH can be found in Table IV. In this comparison, we assume that L-QCH also utilizes the Hadamard design to construct its cyclic quorum system. That is, there will be \( \frac{n-3}{4} \) overlaps in \( n \) slots for each pair of nodes. For each of the rest \( \frac{3(n+1)}{4} \) nonoverlapping slots, the rendezvous probability for two nodes is \( \frac{1}{n} \) because each node selects a channel randomly. Thus, the ETTR of L-QCH is \( \frac{2n^2}{n-1} \). In ACH, \( n \) overlaps exist in \( n^2 \) slots for each pair of nodes and hence the ETTR for ACH is equal to \( n \). In QLCH, a cyclic quorum system generated by Hadamard design \((n,k,\lambda)\) is applied. The performance analysis is very similar to that of RQL being presented above. Two different cases need to be considered for nodes running QLCH: with identical pattern (denoted as \( u \equiv w \)) and with different pattern (denoted as \( u \not\equiv w \)). Since the number of rendezvous for nodes \( u \) and \( w \) is fixed in all the frames (\( \frac{n-1}{n} \) if \( u \equiv w \) and \( \frac{2n^-1}{n} \) if \( u \not\equiv w \)), we have ETTR\(_{u\equiv w} = \frac{2n}{n-1} \) and ETTR\(_{u\not\equiv w} = \frac{2n}{n+1} \). The ETTR for QLCH can be obtained as

\[
P[u \equiv w] \cdot \text{ETTR}_{u\equiv w} + P[u \not\equiv w] \cdot \text{ETTR}_{u\not\equiv w} = \frac{1}{n} \cdot \frac{2n}{n-1} + \frac{2n}{n+1} = \frac{2n}{n^2 - (n+2)}.
\]

The MTTR values for different protocols are calculated in the most conservative way. The frame length of L-QCH is \( n \) while the number of rendezvous is \( \frac{n^2+3}{4n} \), the MTTR of L-QCH is \( (n - \frac{n^2+3}{4n}) + 1 = \frac{3n^2+4n+3}{4n} \). For ACH, the cycle length is \( n^2 \) while the number of rendezvous is \( n \), the MTTR of ACH is \( n^2 - n + 1 \). The frame length of QLCH is \( n \) while when \( u \equiv w \) has a smaller number of rendezvous of \( \frac{n-1}{n} \), so the MTTR of QLCH is \( (n - \frac{n-1}{n} + 1) = \frac{n+3}{2} \).

The comparison of the analytical ETTR and MTTR for different protocols with different channels can be found in Fig. 5 and Fig. 6, respectively. For RQL, the cyclic quorum system \( \Theta(h,k,\lambda) \) is assigned based on Table III. The cyclic LCG system is \( \Omega(\frac{5}{1}, m) \), where \( m \) is the least power of \( 2 \) that is larger than \( h \). For instance, \( m \) is 8, 16 and 32 when the value of \( h \) is 7, 11 and 19, respectively. For reference purposes, we have also written a computer program to obtain the real ETTR and MTTR for RQL, denoted as RQL-real in the figures. From Fig. 5, we can see that RQL is really close to QLCH and outperforms the other two protocols. Note that the MTTR for RQL has a little higher than that of L-QCH and QLCH as shown in Fig. 6. It is the cost of RQL to achieve higher throughput. According to Table IV, MTTR of RQL depends on the value of \( m \) and a smaller \( m \) can be selected to avoid large MTTR. Note that MTTR is the longest possible time interval for two consecutive rendezvous of any pair of nodes. It is just an upper bound. A protocol has a higher MTTR does not imply that it performs worse. In fact, RQL performs much better in terms of system throughput when compared with schemes with smaller MTTR values, as will be verified in the next section.

### Table III

<table>
<thead>
<tr>
<th>( n )</th>
<th>( h )</th>
<th>( \Gamma(h) )</th>
<th>( n )</th>
<th>( h )</th>
<th>( \Gamma(h) )</th>
<th>( n )</th>
<th>( h )</th>
<th>( \Gamma(h) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-6</td>
<td>7</td>
<td>2</td>
<td>31-34</td>
<td>35</td>
<td>7</td>
<td>67-70</td>
<td>71</td>
<td>9</td>
</tr>
<tr>
<td>7-10</td>
<td>11</td>
<td>4</td>
<td>35-42</td>
<td>43</td>
<td>6</td>
<td>71-78</td>
<td>79</td>
<td>10</td>
</tr>
<tr>
<td>11-14</td>
<td>15</td>
<td>3</td>
<td>43-46</td>
<td>47</td>
<td>8</td>
<td>79-82</td>
<td>83</td>
<td>10</td>
</tr>
<tr>
<td>15-18</td>
<td>19</td>
<td>4</td>
<td>47-58</td>
<td>59</td>
<td>5</td>
<td>83-102</td>
<td>103</td>
<td>10</td>
</tr>
<tr>
<td>19-22</td>
<td>23</td>
<td>6</td>
<td>59-62</td>
<td>63</td>
<td>7</td>
<td>103-106</td>
<td>107</td>
<td>10</td>
</tr>
<tr>
<td>23-30</td>
<td>31</td>
<td>4</td>
<td>63-66</td>
<td>67</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table IV

<table>
<thead>
<tr>
<th>Protocol</th>
<th>ETTR</th>
<th>MTTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-QCH</td>
<td>( \frac{4n^2}{n^2+3} )</td>
<td>( \frac{3n^2+4n+3}{4n} )</td>
</tr>
<tr>
<td>ACH</td>
<td>( n )</td>
<td>( n^2 - n + 1 )</td>
</tr>
<tr>
<td>QLCH</td>
<td>( \frac{2(n^2-n+2)}{n^4-1} )</td>
<td>( \frac{n+3}{2} )</td>
</tr>
<tr>
<td>RQL</td>
<td>( 2 + \frac{2(n+m)}{m(n+1)} )</td>
<td>( h + 2\Gamma(h) )</td>
</tr>
</tbody>
</table>

---

\(^5\)The Hadamard difference sets for \( h \leq 107 \) can be found in web site http://hsccl.ntou.edu.tw/hsccl/hds.html.
each slot is set to 0.5 for each channel. For each flow, the source node is randomly selected while one of its neighbors is randomly picked as the destination. Each point in the figures is an average of 20 simulations with each simulating 1000 time slots. The network topology and source-destination pairs were regenerated in each simulation run.

In the following, observations are made from two aspects.

A) Impact of number of flows: In this experiment, different numbers of flows were tested to observe the effect. The number of rendezvous channels was set to 11 while the number of flows was varied from 1 to 60. The throughput comparison for different protocols can be found in Fig. 7. It is obvious that RQL outperforms the others in all scenarios. We believe this is because RQL generates a lot of rendezvous and avoids unnecessary transmissions. Nodes running L-QCH have the lowest throughput because nodes compete for the single rendezvous channel in each frame and produce many transmission collisions.

Fig. 8 shows the TTR comparison for different protocols. We can see that the number of flows have little influence on TTR. QLCH performs the best because it produces the most number of rendezvous. RQL performs a little worse than QLCH because there exist missing frames for nodes running RQL. However, as shown in Fig. 7, QLCH performs worse than RQL because of its lower channel utilization\(^6\). It should be noted that L-QCH performs a little worse than ACH in throughput but outperforms ACH in TTR. This means that using a single rendezvous channel is harmful for throughput but is beneficial for TTR. For ACH, the ETTR is \(n\) which is independent of number of flows.

B) Impact of number of channels: Next, the number of channels was varied from 5 to 13 to observe the influence. The number of flows is fixed at 30 in this experiment. The achieved throughput for different protocols are shown in Fig. 9. As expected, RQL has the best performance in all different cases. The improvement of using RQL is getting larger when more channels are being used. For example, when 7 channels are available, the throughput of RQL is 256\%, 104\%, and 34\% higher than that of L-QCH, ACH, and QLCH, respectively; when 11 channels are available, the throughput of RQL is 453\%, 206\%, and 38\% higher than that of L-QCH, ACH, and QLCH, respectively; when 13 channels are available, the throughput of RQL is 527\%, 209\%, and 48% higher than that of L-QCH, ACH, and QLCH, respectively. L-QCH and ACH perform worse when more channels are available because of the low channel utilization (L-QCH) or high TTR (ACH). On the contrary, QLCH and RQL have higher throughput when more channels are being used because of the steady and low TTR. Again, RQL outperforms QLCH because RQL achieves higher channel utilization.

Fig. 10 shows the TTR comparison for different protocols when different channels are being used. QLCH still has the lowest TTR which is followed by RQL, L-QCH, and ACH. For RQL, QLCH, and L-QCH, the achieved TTR is almost

\(^6\)The channel utilization upper bound of QLCH is 50\% while the channel utilization of RQL may be more than 50\%.
independent of the number of channels being used. In fact, according to our analysis of ETTR listed in Table IV, when the number of channels is large, the ETTR of RQL, QLCH, and L-QCH approaches 2, 2, and 4, respectively. For ACH, the TTR enlarges as the number of channels increases since the ETTR of ACH is $n$.

VI. CONCLUSIONS

Combining the concepts of quorum systems, latin squares, and linear congruential generator, a complete rendezvous guarantee channel hopping protocol for CRNs is proposed in this paper. In the proposed RQL protocol, each node is equipped with only one CR transceiver and individually determines both its channel hopping sequence and its pattern hopping sequence. Each node can obtain other nodes’ channel and pattern schedules if their IDs and time slot offsets are available. We have proved the correctness of RQL and have analyzed its performance. Simulation results verify that RQL performs better than existing L-QCH and ACH protocols in that it achieves lower TTR, higher throughput. And RQL outperforms QLCH protocol in that it achieves higher channel utilization and thus high throughput. We consider that RQL is a practical and efficient protocol which intelligently solves the rendezvous problem in CRNs.

ACKNOWLEDGEMENTS

This research was sponsored by Ministry of Science and Technology, R. O. C., under grant NSC 102-2221-E-019-017-MY3.

REFERENCES


---

**Chih-Min Chao** received the B.S. and M.S. degrees in computer science from Fu-Jen Catholic University and National Tsing-Hua University in 1992 and 1996, respectively, and the Ph.D. degree in computer science and information engineering from National Central University in January of 2004. He was an assistant professor at the TamKang University, Taiwan, from 2004 to 2005. He joined the Department of Computer Science and Engineering, National Taiwan Ocean University where he worked as an assistant professor (2005-2008), an associate professor (2008-2014), and a professor (2014-present). His research interests include mobile computing and wireless communication.

**Hsiang-Yuan Fu** received the B.S. degree in electronic engineering and the M.S. degree in computer science and engineering from the National Taiwan Ocean University in 1988 and 2010, respectively. His research interests include wireless communication and sensor networks.