CACH: Cycle-Adjustable Channel Hopping for Control Channel Establishment in Cognitive Radio Networks

Tsung-Ying Wu[1], Wanjiun Liao[1], and Cheng-Shang Chang[2]
[1]Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
[2]Institute of Communications Engineering, National Tsing-Hua University, Hsinchu, Taiwan
Email: {d96921024, wjliao}@ntu.edu.tw, cschang@ee.nthu.edu.tw

***Abstract*—Establishing control channels in a cognitive radio network (CRN) is an important and challenging problem. To cope with the problem of control channel saturation and the problem of channel blocking by primary users, channel hopping (CH) schemes are commonly used in the literature for control channel establishment in CRNs. There are three metrics that are widely used for evaluating the performance of CH schemes: (i) degree of overlapping (the number of distinct rendezvous channels), (ii) worst case time-to-rendezvous (TTR), and (iii) system load. In this paper, we focus on the symmetric and synchronous setting and propose a novel Cycle-Adjustable Channel Hopping (CACH) scheme that outperforms several existing CH schemes, including SSCH and QCH, in terms of the three metrics. The key idea of CACH is to create an additional layer of logical channels on the top of physical channels so that the cycle of channel hopping sequences can be adjusted to optimize system performance. The mathematic tools for our scheme are based on the operations in Galois fields that are more general than the prime number modular arithmetic used in SSCH. We show that CACH is much more general than SSCH and it can achieve the maximum degree of overlapping while allowing the worst case TTR to be adjustable. It is also much better than QCH in terms of reducing system load while keeping the same degree of overlapping and the same worst case TTR. Our simulation results show that CACH outperforms several existing schemes in many other aspects, including throughput, and robustness to the disturbance of PUs.**

***Keywords*—Cognitive radio, multiple rendezvous, dynamic channel hopping, Galois field**

I. INTRODUCTION

Wireless networks used today are regulated by a fixed spectrum policy. This policy leads to the problem of inefficient usage of radio spectrum [1]. To solve this problem, cognitive radio (CR) [2] was introduced to improve the spectrum efficiency. In a cognitive radio network (CRN), unlicensed users (called secondary users (SUs)) are allowed to use unused licensed spectrum without interfering licensed users (called primary users (PUs)). With the support of software defined radio (SDR) technology, nodes equipped with cognitive radio transceivers (CR transceivers) can intelligently adjust the transmission characteristics (e.g., transmission power, carrier frequency, and modulation strategy) to achieve highly reliable communications and high spectrum efficiency throughout a wide range of spectrum. Therefore, they can quickly switch their operation spectrums and utilize the unused licensed spectrums efficiently.

In a CRN, each SU is associated with a set of channels for communications, and the availability of each channel is determined by the behavior of neighboring PUs. SUs located in different locations may have different available channel sets because their neighboring PUs may be different. In addition, the available channel set of an SU may change with time because the neighboring PUs may change their transmission states. The diverseness of available channel sets makes the problem of establishing a control channel very challenging in a CRN, especially in a fully distributed environment.

The most typical approach for control channel establishment is to use a dedicated global control channel among all SUs [3]-[6]. However, the availability of channel sets among SUs may vary due to the fact that they might have different neighboring PUs. Hence, the likelihood of having a control channel globally available to all SUs is very slim. Even if SUs are able to find a globally available channel, the availability of this dedicated control channel may change over time. When the dedicated control channel is unavailable, the normal operations of SUs may be disrupted. In particular, new data packets cannot be transmitted because the control messages cannot be exchanged even though there are other common available channels. Once a PU starts using its channel, it is very likely that the PU will continue to use this channel for a long time. Thus, all the control messages will be "blocked" during this long duration. Such a problem is known as *the PU long-time blocking problem*. Moreover, using one single control channel may introduce a bottleneck in the operation and may further cause *the control channel saturation problem* in a high node-density environment.

To cope with the control channel saturation problem and the PU long-time blocking problem, channel hopping (CH) schemes are commonly used in the literature [7]-[19]. In a CH scheme, time is usually divided into consecutive *time intervals* and each SU hops to a channel in every time interval according to a specific CH sequence. As discussed in [19], CH schemes can be classified into various categories depending on their assumptions. A CH scheme is called *asymmetric* if one SU can be identified as the *sender* and the other SU can be identified as the *receiver*. For *asymmetric* CH schemes (such as ACH in [18] and ARCH in [19]), the sender and the receiver can use different strategies to rendezvous and thus can achieve better performance than *symmetric* CH schemes (such as SSCH in [7], SYN-MAC in [8], QCH in [9] and DH-MAC in [10]), where both SUs have to follow the same strategy. Also, a CH scheme is *synchronous* if the indices of

time intervals of both SU are the same. Such a synchronous setting can be easily implemented when there is a common GPS clock or there are timing signals from a neighboring PU, e.g., beacons from base stations of local cellular service providers. *Synchronous* CH schemes can achieve better performance than *asynchronous* CH schemes as both SUs know when to start their CH sequences. If clock synchronization is difficult, there are also several novel *symmetric* and *asynchronous* CH schemes that have been proposed in the literature, e.g., SeqR [12], DSREQ [13], CRSEQ [14], ASYNCH-ETCH [15] and JS [16]. A comparison of all these CH schemes can be found in [19].

As addressed in [9] and [19], there are three common metrics for evaluating the performance of a CH scheme: (i) degree of overlapping: the number of distinct channels for two SUs to rendezvous in each operation period, (ii) worst case time-to-rendezvous (TTR) (MTTR in [9]): the maximum time for two SUs to rendezvous (when there is no PU blocking), and (iii) system load: the maximum probability that an SU hops to a particular channel at a particular time interval. Clearly, for the PU long-time blocking problem, a CH scheme should have a large degree of overlapping, preferable the maximum degree of overlapping. On the other hand, to reduce packet delay, it is preferable to have a low worst case TTR. Finally, to mitigate the control channel saturation problem, a CH scheme should minimize its system load so that the average number of SUs that hop to the same channel at the same time interval can be minimized. As pointed out in [9], there is a tradeoff between worst case TTR and system load. In general, one can increase system load to reduce TTR and such a tradeoff can then be used to optimize system performance. Unfortunately, most existing CH schemes [7][8][10]-[16] in the literature were designed for a fixed environment and they cannot be easily adjusted to optimize system performance.

Motivated by this, we focus on the *symmetric* and *synchronous* setting in this paper and propose a novel CH scheme, called Cycle-Adjustable Channel Hopping (CACH) scheme that exploits the tradeoff between system load and worst case TTR to optimize system performance. For this, we first generalize SSCH by proposing the Round-Robin Indemnity-channel Channel Hopping (RRICH) scheme that uses the Galois field operations instead of the prime number modular arithmetic in SSCH. RRICH also achieves the maximum degree of overlapping by implementing "rotation" under the addition field operation. We show that RRICH has the worst case TTR N+1 and the system load $1/(N-1)$ that are the same as those in SSCH. One problem of the RRICH scheme is that the TTR might be very long when the number of channels N is very large. To solve the long TTR problem for a large number of channels N, we further generalize RRICH by proposing CACH, where another layer of logical channels is created. In CACH, SUs rendezvous on logical channels that are in turn mapped to physical channels by using a modulo operation. By so doing, CACH still achieves maximum degree of overlapping. The number of logical channels u could be chosen to be much smaller than the number of physical channels N. As such, the worst case TTR of CACH is reduced to (u+1) at the cost of increasing its system load to $1/u$.

Certainly, choosing the number of logical channels u that optimizes the system performance, e.g., throughput, depends on various system characteristics, e.g., the number of SUs, the number of channels, and the characteristics of PUs. For this, we perform various computer simulations. By setting the number of logical channels u close to the average number of neighbors, our simulation results show that CACH in general can achieve a good throughput that is fairly close to its maximum throughput.

The rest of the paper is organized as follows. We propose RRICH and CACH in Sec. II and Sec. III, respectively. The simulation results are shown in Sec. IV. Finally, the paper is concluded in Sec. V.

II. Round-Robin Indemnity Channel Hopping Sequence

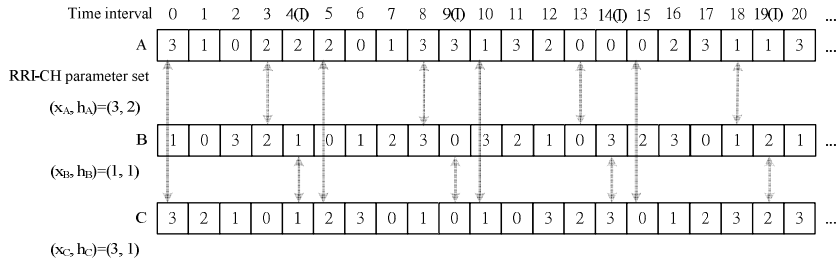
We consider an overlay ad-hoc CRN consisting of both PUs and SUs. There are N non-overlapping orthogonal channels indexed from 0 to N-1. The availability of each channel for each SU is determined by the activity of its neighboring PUs. Therefore, it varies with time and location. Each SU is equipped with *one CR transceiver* and able to accurately detect the availability of the current channel on which it operates. We call this channel the *operation* channel.

As mentioned in the introduction, we focus on the *symmetric* and *synchronous* setting. In such a setting, the time axis is divided into consecutive *time intervals.* We index the time intervals by t=0, 1, 2,…. At the beginning of each time interval, SUs will switch their operation channel according to their predetermined CH sequences. If the channel is available, then SUs can use that channel for information exchange.

Now we introduce our approach for constructing the Round-Robin Indemnity Channel Hopping (RRICH) sequence. Our construction is based on the mathematical theory of Galois fields [20]. A Galois field $GF(N)$ is a set of N elements with two operations $\oplus$ (addition) and $\otimes$ (multiplication) that satisfy various algebraic properties, including the associative law, the commutative law and the distributive law. Moreover, there exists an identity element for addition $\oplus$, called the zero element, and for every element in $GF(N)$, its additive inverse exists. Similarly, there exists an identity element for multiplication $\otimes$, called the one element, and for every nonzero element, its multiplicative inverse exists. Intuitively, we can add, subtract, multiply and divide in a Galois field as in rational numbers.

It is well known that a Galois field $GF(N)$ exists if and only if N is a power of a prime. In particular, if $N=2$, the addition in $GF(2)$ is the exclusive-OR operation and the multiplication in $GF(2)$ is the AND operation. When N is a prime, the addition is the usual addition with the *MOD N* operation and the multiplication is the usual multiplication with the *MOD N* operation. The operations for $GF(2^m)$ are more involved, but they can be easily implemented by using combinatorial logic circuits and have a lot of applications in error correcting codes and network coding.

In RRICH, we assume that N is a prime power. Hence, a Galois field $GF(N)$ with the two operations $\oplus$ and $\otimes$ exists. If N is not a prime power, the solution of this problem will be described later. Denote the N elements in $GF(N)$ as {0,1,2, …, N-1}, where 0 is the zero element (the identity element for $\oplus$) and *1* is the one element (the identity element for $\otimes$). We will use $-a$ to denote the inverse element of a under $\oplus$ and a^{-1} to denote the inverse element of a under $\otimes$. As we can treat these



(a) The hopping sequences for the three SUs

⊕	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

⊗	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

(b) The addition table and the multiplication table for the two operations of Galois field *GF*(4)

Figure 1: RRICH for three SUs with four channels

two operations as usual addition and multiplication, it is well-known that $-(a \oplus b) = (-a) \oplus (-b)$, $a \otimes 0 = 0 \otimes a = 0$ and $a \otimes (-b) = (-a) \otimes b = -(a \otimes b)$ for the Galois field *GF*(*N*).

For each SU *i*, its RRICH sequence is a periodic sequence with period *N*(*N*+1). Each period of *N*(*N*+1) time intervals is called a *frame*. The sequence for SU *i* in a frame is determined by using a set of two CH parameters: $\{x_i, h_i\}$ (called the RRICH parameter set). The parameter x_i is called the *initial seed*. It denotes the initial channel of the RRICH sequence and its value is an integer ranged over [*0, N*-1]. The other parameter h_i is called the *hopping seed*. It is used to determine which channel for SU *i* to switch to. In order for the SU to change its control channel over time, we will not select the zero element as a hopping seed (we note that such a constraint will be removed in CACH in the next section). Hence, the value of a hopping seed is an integer ranged over [1, *N*-1]. Specifically, let $c_i(t)$ be the control channel of SU *i* used at the t^{th} time interval for $0 \le t \le N(N+1)-1$. Suppose that $t = q(N+1)+r$, where *q* is the quotient of *t* divided by *N*+1 and *r* is its remainder. Then $c_i(t)$ is determined by:

$$c_i(t) = \begin{cases} h_i \oplus q, & \text{if } r = N \\ (x_i \oplus q) \oplus (h_i \otimes r), & \text{if } 0 \le r \le N-1 \end{cases}. \quad (1)$$

To see the intuition behind (1), suppose that $0 \le t \le N-1$. For this case, we have $q = 0$. The above equation is simply a "line" in the field *GF*(*N*) with h_i being its "slope." As h_i is a nonzero element, the line is not a constant. As such, every SU hops to different channels as time goes on. Moreover, for two SUs with different hopping seeds, they hop as two lines with different "slopes" and these two lines intersect each other at a unique point. For two SUs with identical hopping seeds, they hop as two "parallel lines." That is why we have to add indemnity time intervals (the time intervals with $r = N$) for them to rendezvous.

For the ease of our presentation, we partition each frame of *N*(*N*+1) time intervals into *N* sub-frames, each with *N+1* time intervals. Specifically, the q^{th} sub-frame, $q = 0, 1, \cdots, N-1$, contains the time intervals from $q(N+1)$ to $q(N+1)+N$. Call the channel that an SU uses at an indemnity interval the indemnity channel. Note that the initial seed and the indemnity channel of an SU are updated by "adding" *q* for every sub-frame while the hopping seed remains unchanged for every sub-frame. As such, the channel selections for every sub-frame behave exactly the same as those in the first sub-frame by re-indexing the channels through a "rotation" under the $\oplus$ operation. That is why we call such a sequence the round-robin indemnity channel hopping scheme.

In Fig. 1(a), we further illustrate how RRICH works by considering an example with three SUs and four channels, i.e., $N = 4$. We show the two operations of *GF*(4) in Fig. 1(b). The indemnity time intervals with $r = N = 4$ are marked with "I" in Fig. 1(a). Suppose that the RRICH parameter sets of SU *A*, SU *B* and SU *C* are initialized to be (3, 2), (1, 1), and (3, 1) respectively. For example, considering the 4th time interval, in this case *q* is equal to 0 and *r* is equal to 4. According to Eq. (1), the control channel of SU *A* is set to $2 \oplus 0$ which is equal to 2. Consider the 7th time interval, where *q* is equal to 1 and *r* is equal to 2. Thus the control channel of SU A is set to $(3 \oplus 1) \oplus (2 \otimes 2) = 2 \oplus 3 = 1$. Thus, the control channel of SU A is 1 at the 7$^{\text{th}}$ time interval.

In the following lemma, we show that an SU can scan all the channels and check the availability of each channel.

Lemma 1. An SU visits all the *N* channels within the first *N* time intervals in each sub-frame.

Proof. Since the hopping seed h_i cannot be the zero element, all the *N* elements in the set $\{(x_i \oplus q) \oplus (h_i \otimes r), 0 \le r \le N-1\}$ are *distinct.* Thus, it is the same as the set $\{0, 1, 2, \ldots, N-1\}$. ■

Our second lemma shows when two SUs rendezvous in each sub-frame.

Lemma 2. Consider two SUs with the parameter sets $\{x_1, h_1\}$ and $\{x_2, h_2\}$.

(i) If they are assigned with the same hopping seed and the same initial seed, i.e., $x_1 = x_2$ and $h_1 = h_2$, then they will rendezvous at each time interval.

(ii) If they are assigned with the same hopping seed ($h_1 = h_2$), but with different initial seeds x_1 and x_2 ($x_1 \ne x_2$), they will rendezvous at the indemnity time intervals (the last

time interval in each sub-frame), i.e., $t = q(N+1) + N$, $q = 0, 1, 2, \ldots, N-1$.

(iii) If they are assigned with different hopping seeds ($h_1 \neq h_2$), they will rendezvous at the r^{th} time interval in each sub-frame, i.e., $t = q(N+1) + r$, $q = 0, 1, 2, \ldots, N-1$, where

$$r = (h_2 \oplus (-h_1))^{-1} \otimes (x_1 \oplus (-x_2)). \quad (2)$$

Proof. The proof for (i) and (ii) follows directly from the rule for the selection of the channels. For (iii), we simply solve the following linear equation:

$$(x_1 \oplus q) \oplus (h_1 \otimes r) = (x_2 \oplus q) \oplus (h_2 \otimes r).$$

To solve this equation, we first add $-(x_2 \oplus q) \oplus (-(h_1 \otimes r))$ on both sides of the identity. Using the algebraic properties of these two operations, one can easily show that

$$(x_1 \oplus (-x_2)) = (h_2 \oplus (-h_1)) \otimes r.$$

Multiplying $(h_2 \oplus (-h_1))^{-1}$ on both sides of the above identity yields

$$r = (h_2 \oplus (-h_1))^{-1} \otimes (x_1 \oplus (-x_2)). \blacksquare$$

In the following theorem, we show that RRICH achieves the *maximum* degree of overlapping if the two SUs do not have the same parameter set. Together with Lemma 1, RRICH achieves the *maximum* degree of overlapping, i.e., all the channels can be used as rendezvous channels for any two SUs in an operation period.

Theorem 3. In RRICH, any two SUs will rendezvous at least once in each sub-frame. Moreover, the channels they rendezvous in the *N* sub-frames are *distinct* if these two SUs do not have the same parameter set.

Proof. From Lemma 2 (i), (ii) and (iii), it is clear that they will rendezvous at least once in each sub-frame. If they are assigned with the same hopping seed ($h_1 = h_2$), but with different initial seeds x_1 and x_2 ($x_1 \neq x_2$), it follows from Lemma 2 (ii) that they will rendezvous in the q^{th} sub-frame on the indemnity channel $h_1 \oplus q$. Thus, the channels they rendezvous in the *N* sub-frames are distinct. On the other hand, if they are assigned with different hopping seeds ($h_1 \neq h_2$), we have from Lemma 2 (iii) that they will rendezvous at the r^{th} time interval in each sub-frame, where *r* is specified in (2). In this case, they will rendezvous on the channel $(x_1 \oplus q) \oplus (h_1 \otimes r)$ in the q^{th} sub-frame and thus the channels they rendezvous in the *N* sub-frames are also distinct (as *r* is not a function of *q*). ■

As a direct consequence of Theorem 3, we show that RRICH also solves the PU long-time blocking problem.

Corollary 4. Suppose that there are only *m* ($m < N$) channels that are used by PUs. Any two SUs will rendezvous within $(m+1)(N+1)$ time intervals.

Proof. If the two SUs are assigned with two different parameter sets, it then follows from Theorem 3 that these two SUs will rendezvous at least $m+1$ times on distinct channels before the end of the m^{th} sub-frame. Since there are only *m* ($m < N$) channels that are used by PUs, these two SUs will rendezvous within $(m+1)(N+1)$ time intervals. On the other hand, if the two SUs are assigned with a common parameter set, then we have from Lemma 2(i) and Lemma 1 that these two SUs will rendezvous at every time interval and on distinct channels in the first *N* time intervals. Thus, they will rendezvous within $m+1$ time intervals. ■

To address the control channel saturation problem, we define the load of a channel at a particular time interval as the *probability* that an SU hops on that channel at that interval. Also, the (maximum) system load is defined as the maximum of the load taken over all channels and all time intervals [9]. Now we show that the system load of RRICH is 1/(*N*-1). For *perfect* load balancing, an SU should hop to every channel at a time interval with an equal probability and the perfect system load should be 1/*N*. For RRICH, we show how we can distribute the traffic into the *N* channels to achieve load balancing. Note that there are *N* choices for the initial seeds and *N*-1 choices for the hopping seeds. To achieve load balancing, we simply assume that each SU chooses its initial seed independently and uniformly over [0, *N*-1] and its hopping seed independently and uniformly over [1, *N*-1]. Let $\{c_{x,h}(t), t = 0, 1, 2, \ldots\}$ be the sequence of the channels generated by using the parameter set $\{x, h\}$. If $t = q(N+1) + N$, then $c_{x,h}(t) = (h \oplus q)$ and an SU that selects such a channel must select the hopping seed *h*. As the hopping seed is selected uniformly over [1, *N*-1], the probability that an SU selects such a channel is then 1/(*N*-1), which is only slightly larger than the ideal load *1/N* when *N* is large. On the other hand, if $t = q(N+1) + r$ for some $0 \leq r \leq N-1$, then for every fixed *h* and every fixed channel *c* there is a unique *x* such that $c = (x \oplus q) \oplus (h \otimes r)$. Thus, the probability that an SU will select channel *c* is the probability that the SU selects the exact *x* that solves the equation. Such a probability is *1/N*. In this case, we achieve perfect load balancing. From both cases, we conclude that the system load of RRICH is 1/(*N*-1), which is the same as the system load of SSCH in Table 1 of [9] . In fact, RRICH is a generalization of SSCH in two folds: (i) RRICH uses the field operations which are much more general than the prime number modular arithmetic in SSCH, and (ii) RRICH implements "rotation" under the $\oplus$ operation and thus the degree of overlapping is *N*, i.e., all the channels. Note that the idea of using "rotation" was previous used in our early conference paper [10]. As a special case of Corollary 4, the worst case TTR for RRICH is *N*+1 time intervals when there is no PU blocking, i.e., $m = 0$. This is also the same as that for SSCH in Table 1 of [9]. In comparison with M-QCH and L-QCH in Table 1 of [9], RRICH has the same degree of overlapping, a lower system load and a larger worst case TTR.

The Maximum Conditional Time To Rendezvous (MCTTR) in [19] is defined as the maximum time for two SU to rendezvous when there are *N-1* blocked channels, i.e., *m=N-1*. From Corollary 4, the MCTTR of RRICH is $N^2 + N$. For the *asymmetric* and *asynchronous* setting, there are CH schemes with MCTTR equal to N^2 [18,19]. As mentioned before, the reason that the MCTTR of the *asymmetric* CH schemes is smaller is because the two SUs can use different strategies. For instance, ARCH in [19] cleverly puts the *N* channels on a ring and has the sender (resp. receiver) walking counterclockwise (resp. clockwise) along the ring. As the sender and the receiver walk in the opposite directions, they work toward each other and thus rendezvous within $N/2$ time intervals (provided that time-parities are the same). The tricky

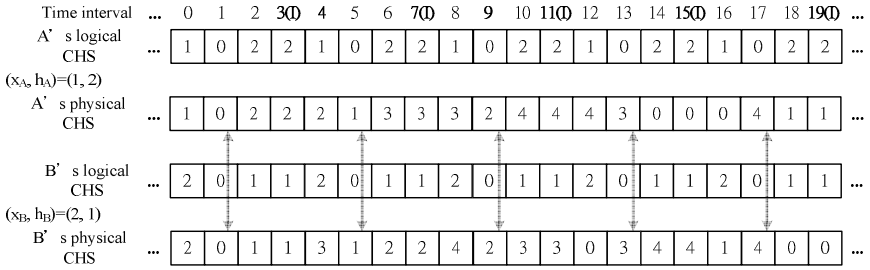


Figure 2: CACH sequences for two SUs with 5 physical channels and 3 logical channels (i.e., N =5 and u=3).

part of such an asymmetric CH scheme is when both SUs would like to become senders at the same time. Then they both walk in the same direction on a ring and will not rendezvous at all. On the other hand, the best MCTTR among all the known *symmetric* and *asynchronous* CH scheme is $N(3N-1)$ in CRSEQ [14], where N is a prime. As the time intervals of the two SUs need not be synchronized in CRSEQ, its MCTTR is much larger than N^2+N in RRICH. To the best of our knowledge, N^2+N seems to be the best MCTTR among all the symmetric and synchronous CHs when the system load is not greater than $1/(N-1)$. In the next section, we will introduce Cycle-Adjustable Channel Hopping (CACH) sequence that can further reduce the MCTTR at the cost of increasing the system load.

In order for the RRICH scheme to work properly, we need the assumption that the number of channels N must be a prime power. If N is not a prime power, we can use a solution which is similar to that proposed in [7]. We can simply choose a prime power N' larger than N. Since $N' > N$, some channels become invalid. When an SU hops to an invalid channel, we could either simply do nothing for that SU at that time interval or remap the channel to one of the N existing channels. Clearly, if we do nothing at that time interval, then this is equivalent to the case that there are $N'-N$ channels blocked by PUs. As such, Corollary 4 is still applicable and we only suffer minor performance degradation. Certainly, the performance can be improved by remapping. However, how to do remapping efficiently requires further study.

III. Cycle-Adjustable Channel Hopping Sequence

One problem of the RRICH scheme is that the time-to-rendezvous (TTR) might be very long when the number of channels N is very large. To solve the long TTR problem for a large number of channels N, we introduce the Cycle-Adjustable Channel Hopping (CACH) scheme in this section. The key idea of CACH is to create another layer of logical channels and have SUs rendezvous on logical channels. By choosing a modulo operation between logical channels and physical channels, CACH still achieves the maximum degree of overlapping as RRICH and thus it can still be used for solving the PU long-time blocking problem.

To reduce the TTR, we choose a much smaller prime power u for the construction of the first sub-frame in RRICH and have two SUs rendezvous on one of the u logical channels in the first $u+1$ time intervals. As in the construction of RRICH, we find a Galois field $GF(u)$ with the two operations $\oplus$ and $\otimes$. Then SU i chooses its parameter set $\{x_i, h_i\}$, where x_i is the initial seed and h_i is the hopping seed. However, unlike RRICH, both the initial seed and the hopping seed are chosen in [0,u-1]. In other words, the hopping seed can be the zero element in the $GF(u)$ used here. Each CACH sequence is a periodic sequence with period $(u+1)N$. For $0 \le t \le (u+1)N-1$, we further partition this period of $(u+1)N$ time intervals into N sub-frames, each with $(u+1)$ time intervals. The last interval in a sub-frame is called the indemnity time interval. Let $\ell_i(t)$ and $c_i(t)$ be the logical channel and the physical channel used by SU i at the t^{th} time interval. Suppose that $t = q(u+1)+r$, where q is the quotient of t divided by $(u+1)$ and r is its remainder. Then $\ell_i(t)$ and $c_i(t)$ are determined by the following equation:

$$\ell_i(t) = \begin{cases} h_i, & \text{if } r = u \\ x_i \oplus (h_i \otimes r), & \text{if } 0 \le r \le u-1 \end{cases},$$

$$c_i(t) = (\ell_i(t) + q) \mod N. \qquad (3)$$

The construction of the sequence $\{\ell_i(t), t \ge 0\}$ is the same as that in (1) except we remove the effect of q. Thus, the sequence $\{\ell_i(t), t \ge 0\}$ is a periodic sequence with period $(u+1)$ and it repeats itself in every sub-frame. The index q is used in the mapping from a logical channel to a physical channel through the modulo operation in (3). As such, the physical channels used in each sub-frame are different.

Fig. 2 gives an example for the construction of CACH sequences for N=5 and u=3. In this example, the addition in $GF(3)$ is the usual addition with the *MOD 3* operation and the multiplication in $GF(3)$ is the usual multiplication with the *MOD 3* operation. Since u=3, each sub-frame contains four time intervals with the last time interval in each sub-frame being the indemnity interval. For SU A with parameter set $(x_A, h_A) = (1, 2)$, we have $\ell_A(0) = x_A = 1$, $\ell_A(1)=(x_A+h_A) \mod 3=0$, and $\ell_A(2)=(x_A+2h_A) \mod 3=2$. As the last time interval in each sub-frame is the indemnity interval, $\ell_A(3) = h_A = 2$. Note that the logical channel hopping sequence $\ell_A(t)$ repeats itself in each sub-frame with the sequence 1, 0, 2, 2. The physical channel hopping sequence $c_A(t)$ then adds 1 with the *MOD 5* operation to 1, 0, 2, 2 in each sub-frame and that leads to 1, 0, 2, 2 for the 0^{th} sub-frame, 2, 1, 3, 3 for the 1^{st} sub-frame, 3, 2, 4, 4 for the 2^{nd} sub-frame, 4, 3, 0, 0 for the 3^{rd} sub-frame and 0, 4, 1, 1 for the 4^{th} sub-frame. Both the logical channel hopping sequence and the physical channel hopping sequence for SU B with the parameter set $(x_B, h_B) = (2, 1)$ are also shown in Fig 2.

Following the same argument as in the proof of Lemma 2, we have the following lemma for CACH. Note that this lemma

still holds even though we allow the hopping seed to be the zero element.

Lemma 5. Consider two SUs with the parameter sets $\{x_1, h_1\}$ and $\{x_2, h_2\}$.

(i) If they are assigned with the same hopping seed and the same initial seed, i.e., $x_1 = x_2$ *and* $h_1 = h_2$, then they will rendezvous at each time interval.

(ii) If they are assigned with the same hopping seed ($h_1 = h_2$), but with different initial seeds x_1 and x_2 ($x_1 \neq x_2$), they will rendezvous at the indemnity time intervals (the last time interval in each sub-frame), i.e., $t = q(u+1) + u$, $q = 0, 1, 2, \ldots, N-1$.

(iii) If they are assigned with different hopping seeds ($h_1 \neq h_2$), they will rendezvous at the r^{th} time interval in each sub-frame, i.e., $t = q(u+1) + r$, $q = 0, 1, 2, \ldots, N-1$, where

$$r = (h_2 \oplus (-h_1))^{-1} \otimes (x_1 \oplus (-x_2)). \tag{4}$$

In Theorem 6, we show that CACH also achieves the maximum degree of overlapping as RRICH.

Theorem 6. Any two SUs will rendezvous at least once in each sub-frame. Moreover, the physical channels they rendezvous in the first *m* sub-frames contain at least *m distinct* channels, $m = 1, 2, \ldots, N$.

Proof. From Lemma 5 (i), (ii) and (iii), it is clear that they will rendezvous at least once in each sub-frame. If they are assigned with the same hopping seed ($h_1 = h_2$), it follows from Lemma 5 (i) and (ii) that they will rendezvous in the q^{th} sub-frame on the logical indemnity channel h_1 and thus on the physical channel $(h_1 + q) \bmod N$. It is clear that all the elements in the set $\{(h_1 + q) \bmod N, q = 0, 1, 2, \ldots, m-1\}$ are distinct. On the other hand, if they are assigned with different hopping seeds ($h_1 \neq h_2$), we have from Lemma 5 (iii) that they will rendezvous at the r^{th} time interval in each sub-frame, where *r* is specified in (4). In this case, they will rendezvous on the logical channel $x_1 \oplus (h_1 \otimes r)$ in the q^{th} sub-frame and on the physical channel $((x_1 \oplus (h_1 \otimes r)) + q) \bmod N$. Once again, it is clear that all elements in set $\{((x_1 \oplus (h_1 \otimes r)) + q) \bmod N, q = 0, 1, ..., m-1\}$ are distinct. ■

Analogous to proof for Corollary 4, one can use the results in Theorem 6 to show that CACH also solves the PU long-time blocking problem.

Corollary 7. Suppose that there are $m\,(m < N)$ channels that are used by PUs. Any two SUs will rendezvous within $(m+1)(u+1)$ time intervals.

In comparison with RRICH, the worst case TTR for CACH is shorter than that of RRICH if $u < N$. However, this is at the cost of increasing the system load and thus the possibility of causing the control channel congestion. To see this, recall that the load of a channel at a particular time interval is defined as the probability that an SU hops on that channel at that interval. Note that there are *u* choices for the initial seeds and *u* choices for the hopping seeds in CACH. As in RRICH, we simply assume that each SU chooses its initial seed and its hopping seed independently and uniformly over [0,*u*-1]. Thus, each SU is distributed uniformly to one of the *u* logical channels in each time interval. Thus, the probability that an SU is distributed in a logical channel (and the corresponding physical channel) is simply $1/u$, which could be substantially higher than the ideal load $1/N$ when $N >> u$. As a direct consequence of Corollary 7, the MCTTR of CACH is $N(u+1)$ which could be substantially smaller than N^2 in ACH [18] and ARCH [19]. We also note that it is difficult to create an additional layer of logical channels in the *asynchronous* setting, where the two SUs do not have the same indices of time intervals.

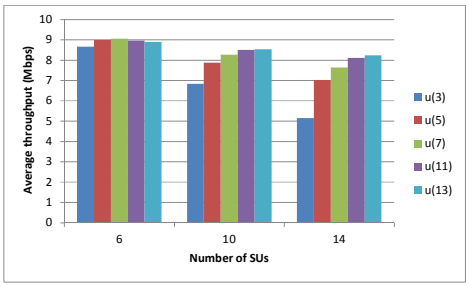
In particular, if we choose *u=2*, the worst case TTR for CACH is 3 and its system load is only 1/2, which is lower than 2/3 of M-QCH [9]. For this case, CACH is better than M-QCH as CACH has a lower system load while keeping the same degree of overlapping and the same worst case TTR. Now we compare CACH with L-QCH [9]. If the maximum allowable TTR is τ, it is shown in Theorem 2 of [9] that the system load of any QCH system is at least $1/\sqrt{\tau}$. L-QCH is the QCH system with the system load $1/\sqrt{\tau}$. Taking $u = \tau - 1$, we then derive that the system load of the CACH scheme is only $1/(\tau - 1)$, which is significantly lower than $1/\sqrt{\tau}$ in L-QCH. In view of these, we conclude that CACH is in general much better than QCH in terms of reducing system load while keeping the same degree of overlapping and the same worst case TTR.

Certainly, choosing the number of logical channels *u* that optimizes the system performance, e.g., throughput, depends on various system characteristics, e.g., the number of SUs, the number of channels, and the characteristics of PUs. For this, we will perform various computer simulations in the next section. One of our findings from the simulations results is to set the number of logical channels *u* close to the average number of neighbors for a reasonably good throughput. As the load of CACH is *1/u*, the average number of SUs that hop to a rendezvous channel is close to 1 if *u* is close to the average number of neighbors.

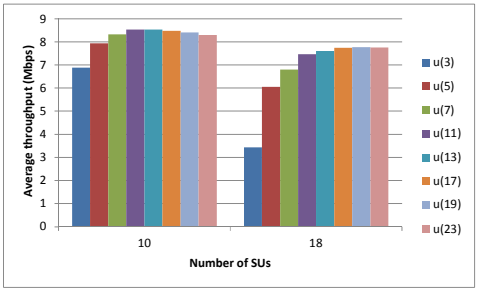
Note that setting *u* to be close to the average number of neighbors depends on the topology of the network and might be difficult to implement in a *dynamic* network. If we view the average number of SUs hopping to a rendezvous channel as the *utilization* of the rendezvous channel, then the key factor for achieving a good throughput is to maintain a reasonably good utilization in a rendezvous channel. Instead of directly estimating (or tracking) the average number of neighbors, one might estimate the average number of contentions in a rendezvous channel and use that to maintain a reasonably good utilization in a rendezvous channel. Specifically, we can partition the average number of contentions in a rendezvous channel into several levels and associate each level with an appropriate *u*. By so doing, we can make CACH adapt to the dynamic change of the network. However, how to estimate the number of contentions in a rendezvous channel and establish the mapping between that and *u* require further study

IV. Simulation Results

In this section, we perform various computer simulations via an event-driven C++ simulator. In our simulations, we only consider disjoint flows, where each source SU and each destination SU cannot have multiple flows. However, we do allow SUs to change its hopping seeds. Specifically, we let

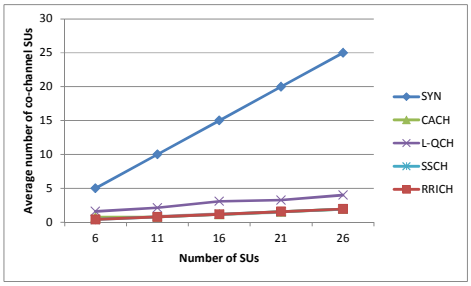


(a) The number of channels is 13.

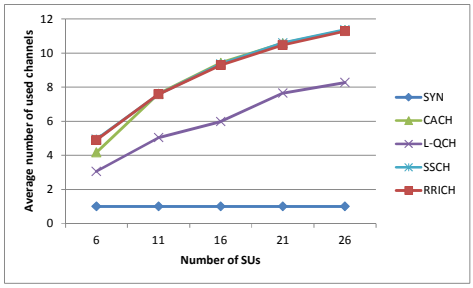


(b) The number of channels is 23.

Figure 3: The throughputs of CACH for different number of logical channels.



(a) Effect of the number of SUs on the average number of co-channel SUs.



(b) Effect of the number of SUs on the average number of used channels.

Figure 4: Effects of the number of channels on the average number of co-channel SUs and the average number of used channels.

each SU change its hopping seeds with probability 1/100 at the beginning of each time interval. These flow pairs and PUs are distributed randomly in a region. The transmission range and the interference range are set to 250m and 550m, respectively. The behavior of a PU is modeled by a two-state ON-OFF Markov chain, where both ON periods and OFF periods are independent, and exponentially distributed. The simulation time is set to 100s. The time interval is set to 6 ms. In the first 2 ms, each pair of SUs will exchange the control messages. If they correctly receive the control messages from each other, they can use the control channel as their data channel for transmitting data in the following 4ms. All SUs content the right of access channel through IEEE 802.11 Distributed Coordination Function.

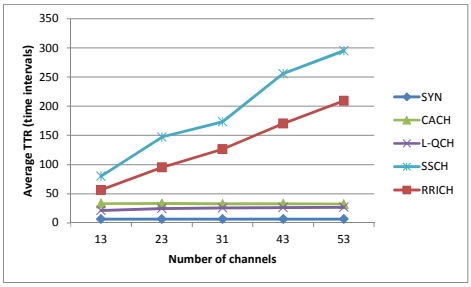
A. *Selection of the Number of Logical Channels*

In this section we address the problem for the selection of the number of logical channels in CACH. We consider that there are some SUs distributed in a 380mx380m region. As such, each SU is a neighbor of another SU. Each channel is associated with a PU. The mean of OFF periods is set to 10s. The mean of ON periods is set to 10s with probability 1/5 and is set to 30s with probability 4/5. The channel capacity is set to 54Mbps and the source SU always has packets to send. In Fig. 3(a), we show the throughputs of CACH when the number of channels N is set to 13. In Fig. 3, $u(x)$ means the number of logical channels u is set to x. When the number of SUs is set to 6 (resp. 10 and 14), CACH has the largest throughput if the number of logical channels u is set to 7 (resp. 13 and 13). It can be observed that when the number of logical channels is set to the value close to the number of neighbors, CACH can achieve a good throughput. When the difference between the number of logical channels and the number of neighbors increases, the throughput decreases. In Fig. 3(b), we consider another setting with the number of channels N being set to 23. When the number of SUs is set to 10 (resp. 18), CACH has the largest throughput if the number of logical channels u is set to 11 (resp. 19). The results in Fig. 3(b) also show that one should choose the number of logical channels close to the number of neighbors.

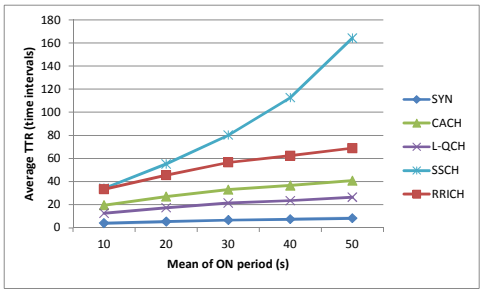
Note that the performance of throughput might be quite sensitive to the selection of the number of logical channels u in some cases. As discussed before, if we view the average number of SUs hopping to a rendezvous channel as the utilization of the rendezvous channel, then the key factor for achieving a good throughput is to maintain a reasonably good utilization in a rendezvous channel. As shown in Fig. 3, if we set u too small, then the utilization of a rendezvous channel is too high and that leads to lots of collisions and sharp degradation of throughput.

B. *Performance comparisons without data traffic*

Next, we compare the performance of SYN, SSCH, L-QCH, RRICH and CACH for various performance metrics, including the average number of co-channel SUs (for a particular SU per time interval), the average number of used channels (per time interval), and the average TTR. In order to see the effects (and the insights) of these rendezvous algorithms, we do this without introducing data traffic, i.e., no data flows. We first consider the case where six SUs are distributed randomly in a 380mx380m region. Also, the SUs do not change their CH parameters. In this simulation, there are 13 channels and each channel is associated with a PU. The mean OFF period of a PU is set to 10s. The mean of ON periods is set to 10s with probability 1/5 and is set to 30s with probability 4/5. In Fig. 4(a), we show the effect of the number of SUs on the average number of co-channel SUs (for a particular SU). A co-channel SU for a particular SU in a particular time interval is an SU that operates on the same channel as that SU in the time interval. To measure this, we choose an arbitrary SU and take the average of the number of



(a) Effect of the number of channels on average TTR.



(b) Effect of the PU behavior on the average TTR.

Figure 5: Effects of the number of channels and PU behavior on the average TTR.

co-channel SUs over time. Clearly, the larger the average number of co-channel SUs is, the larger the co-channel interference is. As such, a large average number of co-channel SUs might suffer from the problem of control channel saturation. It can be seen from Fig. 4(a) that both RRICH and SSCH have the same average number of co-channel SUs. When the number of SUs is larger than 11, CACH is reduced to RRICH as the number of logical channels is upper bounded by the number of channels. Also, the average number of co-channel SUs for L-QCH is larger than those of RRICH, CACH, and SSCH. Such a result is expected as the system load of L-QCH is larger than the system loads of RRICH, CACH, and SSCH. Since the system load of SYN is 1, the average number of co-channel SUs for SYN increases *linearly* with respect to the number of SUs. On the other hand, the average numbers of co-channel SUs for RRICH, CACH, SSCH, and L-QCH only increase slowly with respect to the increase of the number of SUs. In Fig. 4(b), we further show the effect of the number of SUs on the average number of *used* channels in a time interval. A channel is said to be *used* in a time interval if there is (at least) one SU that operates that channel as the control channel in that time interval. To measure this, we count the number of used channels in every time interval and then take its average. Intuitively, a rendezvous algorithm that has a large average number of used channels tends to distribute its traffic evenly over the channels. It is observed that RRICH, CACH and SSCH have same average number of used channels when the number of SUs is larger than 11. When the number of SUs is only 6, the average number of used channels for RRICH is larger than that of CACH because the system load in CACH is larger than that of RRICH. Also, the average number of used channels for CACH is better than L-QCH, even when the number of SUs is 6. This is because L-QCH only distribute the control traffic over the time (but not over the channels).

In Fig. 5(a), we show the effect of the number of channel on the average TTR. Since SYN allows all SUs to hop to the same channel, it has the lowest average TTR. Since the worst case TTRs of CACH and L-QCH are independent of the number of channels, their average TTRs are also not influenced by the number of channels. However, the worst case TTRs of RRICH and SSCH depend on the number of channels, and the average TTRs of RRICH and SSCH increases when the number of channels increases. Moreover, since the degree of overlapping of RRICH is N, RRICH has a lower average TTR than SSCH (as SSCH suffers from the PU long-time blocking problem). In Fig. 5(b), we measure the effect of the mean ON period (of a PU) on the average TTR. When the mean ON period of each PU is set to 10s, RRICH and SSCH have the same average TTR. However, when the mean of the ON period is increased, the average TTR of SSCH is increased rapidly due to the long-time PU blocking problem. The other channel hopping schemes do not increase their average TTRs quickly because their degree of overlapping are equal to N and then they are immune to the long-time PU blocking problem. In view of Fig. 5(a) and Fig. 5(b), we note that L-QCH has a lower average TTR than that of CACH. This is because that L-QCH has a heavier system load than CACH. On the other hand, CACH have a smaller number of co-channel SUs and a larger number of used channels than those of L-QCH.

C. *Throughput*

In this section, we compare the throughputs of RRICH and CACH (via simulations) with other existing schemes, including SSCH [7], SYN-MAC [8] and L-QCH [9]. In the simulation, the number of logical channels u of CACH is set to the largest prime number that is smaller than the average number of neighbors of the SUs. In Fig. 6(a), we show the effect of the number of channels on the average throughput (per flow). In order to simulate the scenario with heterogeneous available channel sets, we distribute 10 SUs in a 1000mx1000m region independently and uniformly. We then generate 100 network topologies and calculate the average number of neighbors of a SU. In this case, each SU has 5 neighbors on average. Each channel is associated with 2 PUs. The mean of OFF periods is set to 10s. The mean of ON periods is set to 10s with probability 1/5 and is set to 30s with probability 4/5. The channel capacity is set to 54Mbps and the source SU always has packets to send. The packet arrival process is modeled by a Poisson process.

It is observed that SYN-MAC has the highest average throughput because SYN-MAC has almost no contention in this scenario and the lowest worst case TTR. The average throughputs of CACH and L-QCH are not affected by the increase of the number of channels N because their worst case TTRs do not depend on N. Although both L-QCH and CACH have the same worst case TTR, CACH still has a higher average throughput. This is because CACH distributes control messages not only over the time line but also over all the channels (i.e., multiple rendezvous) and L-QCH only distributes control messages over the time line. When the number of channels N increases, the average throughput of RRICH decreases quickly because of the long TTR problem. Note that SSCH has the poorest performance because SSCH does not achieve the maximum degree of overlapping. In Fig. 6(b), we show the effect of the number of SUs on the average throughput. The number of channels N is set to 13. When there

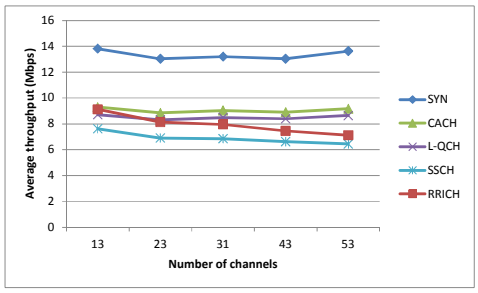
are 10, 20, 30, 40, and 50 SUs distributed randomly in a 1000mx1000m region, the average number of neighbors of an SU is 5, 11, 16, 22 and 28 respectively. When the number of neighbors is larger than 10, the number of logical channels of CACH is set to the number of channels. Hence, CACH is reduced to RRICH and their average throughputs are almost the same when the number of SUs is larger than 20 in Fig. 6(b). When the number of SUs increases, the average throughput of SYN-MAC decrease much rapidly than the other schemes. This is because SYN-MAC now suffers from the control channel saturation problem. Since RRICH, CACH, L-QCH and SSCH do not have the control channel saturation problem, their average throughputs decrease slowly with the same slope as shown in Fig. 6(b).

V. CONCLUSION

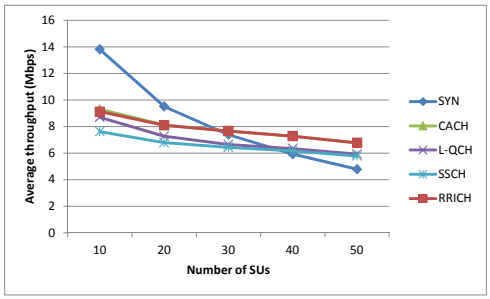
In this paper, we proposed a novel Cycle-Adjustable Channel Hopping (CACH) scheme for control channel establishment in cognitive radio networks. For this, we first extended SSCH to RRICH by introducing Galois fields and rotating rendezvous channels. The key idea of CACH is the creation of another layer of logical channels in RRICH so that the TTR in CACH can be adjusted to optimize system performance. We showed that CACH is better than both M-QCH and L-QCH in terms of minimizing the system load while maintaining the same degree of overlapping and the same worst case TTR. Via extensive computer simulations, it is suggested that the number of logical channels in CACH should be close to the average number of neighbors to obtain a good throughput. Also, the simulations results show that CACH outperforms existing schemes in many aspects, including throughput, and robustness to the disturbance of PUs.

ACKNOWLEDGEMENT

This work was supported in part by the Excellent Research Projects of National Taiwan University, under Grant Number AE00-00-04, and in part by National Science Council (NSC), Taiwan, under Grant Numbers NSC102-2221-E-002-014-MY2 and 102-2221-E-007 -006 -MY3.



(a) Effect of the number of channels on throughput.



(b) Effect of the number of SUs on throughput.

Figure 6: Effects of the number of channels and the number of SUs on throughput.

REFERENCES

[1] Federal Comm. Commission, "Spectrum Policy Task Force Report," Washington, DC, *FCC 02-155*, 2002.

[2] J. Mitola III and G. Q. Maguire Jr.,"Cognitive Radio: Making Software Radios More Personal," *IEEE Personal Communications*, Aug. 1999.

[3] J. Zhao, H. Zheng and G.-H. Yang, "Distributed Coordination in Dynamic Spectrum Allocation Networks," in *Proc. IEEE DySPAN'05*.

[4] L. Le and E. Hossain, "OSA-MAC: A MAC Protocol for Opportunistic Spectrum Access in Cognitive Radio Networks," in *Proc. IEEE WCNC'08*.

[5] T. Chen et al. "CogMesh: A Cluster-based Cognitive Radio Network," in *Proc. IEEE DySPAN'07*.

[6] X. Zhang and H. Su, "CREAM-MAC: Cognitive Radio-EnAbled Multi-Channel MAC Protocol Over Dynamic Spectrum Access Networks," *IEEE Journal on Selected Topics in Signal Processing*, Vol. 5, No. 1, pp. 110-123, February 2011.

[7] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-*Hoc Wireless Net*works," in *Proc. ACM MobiCom'04*.

[8] Y. R. Kondareddy and P. Agrawal, "Synchronized MAC Protocol for Multi-hop Cognitive Radio Networks," in *Proc. IEEE ICC'08.*

[9] K. Bian, J.-M. Park, and R. Chane, "A Quorum-based Framework for Establishing Control Channels in Dynamic Spectrum Access Networks," in *Proc. ACM MobiCom'09*.

[10] C.-F. Shih, T,-Y, Wu, and W. Liao, "DH-MAC: A Dynamic Channel Hopping MAC Protocol for Cognitive Radio Networks," in *Proc. IEEE ICC'10*.

[11] H.-S. W. So, G. Nguyen, J, Walrand, "Practical Synchronization Techniques for Multi-Channel MAC," in *Proc. ACM MobiCom'06*.

[12] L. DaSilva and I. Guerreiro, "Sequence Based Rendezvous for Dynamic Spectrum Access," in *Proc. IEEE DySPAN'08.*

[13] D. Yang, J. Shin, and C. Kim, "Deterministic Rendezvous Scheme in Multichannel Access Networks," *Electronics Letters*, Vol. 46, No. 20, pp. 1402-1404, 2010.

[14] J. Shin, D. Yang, and C. Kim, "A Channel Rendezvous Scheme for Cognitive Radio Networks," *IEEE Communications Letter*, vol. 14, no. 10, pp. 954-956, 2010.

[15] Y. Zhang, Q. Li, G. Yu, and B. Wang, "ETCH: Efficient Channel Hopping for Communication Rendezvous in Dynamic Spectrum Access Networks," in *Proc. IEEE INFOCOM'11*.

[16] Z. Lin, H. Liu, X. Chu, and Y.-W. Leung, "Jump-Stay Based Channel-hopping Algorithm with Guaranteed Rendezvous for Cognitive Radio Networks," in *Proc. IEEE INFOCOM'11*.

[17] N. C. Theis, R. W. Thomas, and L. A. DaSilva, "Rendezvous for Cognitive Radios," *IEEE Transactions on Mobile Computing*, Vol. 10, No. 2, pp. 216--227, 2011.

[18] K. Bian and J.-M. Park. "Asynchronous channel hopping for establishing rendezvous in cognitive radio networks." In *Proc. IEEE. INFOCOM,* 2011.

[19] G.-Y. Chang, W.-H. Teng, H.-Y. Chen, and J.-P. Sheu, "Novel Channel-Hopping Schemes for Cognitive Radio Networks," *IEEE Transactions on Mobile Computing*.

[20] R. P, Grimaldi, Discrete and Combinational Mathematics: An Appl*ied Introduction. Addison* Wesley 2004.