Spectrum Allocation With Guaranteed Rendezvous in Asynchronous Cognitive Radio Networks for Internet of Things

Sulagna Mohapatra, Prasan Kumar Sahoo, *Senior Member, IEEE*, and Jang-Ping Sheu, *Fellow, IEEE*

***Abstract*—The massive usage of Internet-of-Things devices in various smart applications enables spectrum scarcity issues. In order to enhance the dynamic spectrum capability, cognitive radio network (CRN) is considered as a key technology to address the spectrum scarcity problem. However, the establishment of a common communication channel in CRN by considering the unlicensed heterogeneous devices in an asynchronous environment is a challenging problem. In this paper, a novel asymmetric asynchronous channel hopping mechanism is designed, where secondary users have different sets of available channels and can enter into the network without any global clock synchronization. The proposed algorithms can guarantee the rendezvous within a small interval of time with minimum inter rendezvous intervals. Simulation results show that the designed protocol outperforms over the existing channel hopping algorithms in terms of the degree of rendezvous, average time to rendezvous and throughput.**

***Index Terms*—Asymmetric, asynchronous, channel hopping (CH), cognitive radio networks (CRNs), Internet of Things (IoT), spectrum sharing.**

I. Introduction

THE INNOVATIVE Internet of Things (IoT) technology embedded with various smart applications constitutes a network of interconnected things, such as sensors and actuators equipped with different communication interfaces. However, some portion of the allocated spectrum is over-utilized in the smart IoT application environment, whereas some part of it is under-utilized as the IoT devices transmit their data at different instants of time based on the application requirements [1]. For example, a smart restaurant embedded with various smart IoT devices, such as smart light, smart water meter, and smart chairs are preallocated with certain number of licensed wireless channels. The light, fire, and smart chair sensors transmit their data very frequently making the allocated spectrum or channels overcrowded. In contrast, the sensors like weather or water meter sensors transmit the data in certain intervals of time, which makes the allocated spectrum under-utilized and therefore creates the spectrum holes. In this scenario, the light sensor or the door sensor can utilize the vacant portion of the weather sensor enabling uniform utilization of the spectrum. This solution domain can only be possible with the use of cognitive radio network (CRN) [2].

Cognitive radio (CR) has evolved as a promising technology to use the allocated spectrum efficiently in IoT application environments. Federal communications commission has introduced the CRN technology, which allows the unlicensed secondary users (SUs) to utilize the unused spectrum assigned to the licensed primary users (PUs) and therefore enhances the dynamic spectrum access [1], [3]–[5]. The main objective of the CRN is to make utilization of the unused spectrum by fairly distributing it among all the SUs in absence of the PUs [6]–[8]. Generally, PUs have the higher priority to utilize the spectrum without interference. However, the unlicensed SUs are allowed to opportunistically exploit the unused licensed spectrum called as spectrum holes [6] in absence of the PUs. Each SU is equipped with one or more CRs to sense the signals of the PUs periodically within the spectrums to find out the unutilized available channels, which may vary among the SUs [9].

Unlike CRN, other wireless communication protocols, such as ZigBee, Wi-Fi, Bluetooth, and Cellular network do not have intelligent CRs to sense the vacant spectrum. Hence, it leads to the nonuniform utilization of the allocated spectrum. Furthermore, the authorized IoT devices operated on those traditional wireless technologies do not have the capacity to share the allocated spectrum with other users even it is unused, which can be considered as a major pitfall in traditional and cellular wireless communication technologies [5]. However, the concept of CRN can be used with the IoT communication protocols, such as ZigBee, Wi-Fi, and Bluetooth for efficient utilization of the allocated spectrum dynamically as they operate on the same 2.4 GHz industrial, scientific, and medical (ISM) radio band. To fulfill the huge and frequent bandwidth demands of the mobile devices, combination of CRN with 5G technology is proposed. The use of CRN with 5G technology can allow the licensed users of one mobile network to use the vacant spectrum of another mobile company during the congestion or network over-utilization.

Manuscript received July 15, 2018; revised August 29, 2018; accepted September 16, 2018. Date of publication September 27, 2018; date of current version July 31, 2019. This work was supported by the Ministry of Science and Technology, Taiwan, under Grant MOST 106-2221-E-007-019-MY3, Grant 106-2221-E-182-014, and Grant 107-2221-E-182-073. *(Corresponding author: Prasan Kumar Sahoo.)*

S. Mohapatra is with the Department of Computer Science and Information Engineering, Chang Gung University, Taoyuan 33302, Taiwan (e-mail: d0521007@stmail.cgu.edu.tw).

P. K. Sahoo is with the Department of Computer Science and Information Engineering, Chang Gung University, Taoyuan 33302, Taiwan, and also with the Division of Colon and Rectal Surgery, Chang Gung Memorial Hospital, Linkou 33305, Taiwan (e-mail: pksahoo@mail.cgu.edu.tw).

J.-P. Sheu is with the Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan (e-mail: sheujp@cs.nthu.edu.tw).

Digital Object Identifier 10.1109/JIOT.2018.2872459

Apart from the advantages of efficient spectrum utilization, the need for integration of CRN with the IoT is due to the following reasons [5]. The traditional wireless technologies, such as Bluetooth, Wi-Fi, and ZigBee are operated on ISM band and support maximum distance up to 100 m. Hence, these legacy wireless technologies are not suitable for the IoT applications, such as smart grid, smart vehicular transmission, and environmental monitoring, where the deployed smart IoT devices need to communicate with far distance base stations. Furthermore, those traditional wireless technologies do not have any smart sensing or switching capability in case of any external obstruction during the data transmission. However, this limitation can be overcome by using CRN-based IoT paradigm, where the long distance communication can be achieved with help of the SUs as they have smart cognitive antennas, which can detect the neighboring SUs as well as the vacant channels in case of any congestion.

The fare spectrum distribution quality of CRN as well as the operation in ISM band, avoid the cost of extra bandwidth purchase. In a smart IoT application environment, the IoT devices are embedded with multiple heterogeneous wireless communication technologies, which cause the interference problem upon mobility of the devices from one place to another. However, the smart SUs can search the interference-free channels efficiently in CRN enabled IoT environment, which can provide better collision-free communication. Furthermore, the novel CRN technology incorporates the seamless service to the Internet enabled IoT devices in any place on the go due to its compatibility with other wireless technologies. However, IoT devices embedded with traditional wireless technologies may suffer from noncontiguous communication as the embedding wireless mechanism may not be available at every place.

In CRN, SUs can communicate with each other through the process of rendezvous [10], [11]. They need to meet with each other in the same available channel at the same time to establish the rendezvous, which includes both the process of negotiation and data transfer. In general, rendezvous process of SUs in CRN is carried out by channel hopping (CH) approaches [9], [12], [13]. Each SU hops from one channel to another to make its rendezvous successful [3], [14] by using the CH sequences. In this approach, the network is considered as either synchronous or asynchronous. All SUs normally enter into the network at the same time in the synchronous environment, whereas SUs arrive at different time instances without any clock synchronization [3], [6], [15]–[18] in asynchronous scenario. The network model is said to be symmetric, if all the SUs have same set of available channels; otherwise it is said to be asymmetric, where SUs have different sets of available channels [6], [15], [16] with at least one common available channel for the rendezvous.

Following the CH approach, performance of the SUs is mainly evaluated based on two important metrics, such as the degree of rendezvous and time to rendezvous (TTR) [6]. Generally, the degree of rendezvous is defined as how many times two SUs rendezvous with each other in a CH sequence (CH_Seq). The TTR of an SU is explained as the number of time slots required for an SU to rendezvous with another SU. The worst case of TTR is called as the maximum TTR (MTTR). Generally, the smart IoT devices are operated in an asymmetric asynchronous environment, where each IoT device is equipped with heterogeneous number of channels having different data transmission time. In this paper, a novel CH algorithm called asymmetric asynchronous CH (AACH) is proposed to enable the guaranteed rendezvous within a short period of time, which is very challenging in the asymmetric asynchronous environment.

The rest of this paper is organized as follows. Related works with contributions are presented in Section II. System model of the proposed CH protocol is given in Section III. The proposed asymmetric and asynchronous CH protocol is described in Section IV. Analysis of different network metrics is given in Section V. Performance evaluation of the protocol is made in Section VI. Concluding remarks with future works are presented in Section VII.

II. Related Work

Recently, many researchers focus on various CH approaches for the communication in CRN considering the asymmetric asynchronous environment. The symmetric asynchronous rendezvous CH (SARCH) scheme proposed in [3] can bring rendezvous between the sender and receiver though they have different sets of available channels. Any SU can act as a sender or receiver flexibly in SARCH using the same generated CH_Seq without any preassigned role. To achieve the guaranteed rendezvous in the asymmetric scenario, authors have introduced SA-adaptive CH sequences by replacing each unavailable channel of an SU with its available channel in its SA-default sequence. However, the degree of rendezvous is very less and is nonuniform. Besides, the MTTR value is large enough, i.e., $2(2N + 1)$.

In [11], dynamic asymmetric-role quorum-based CH (D-QCH) and symmetric-role quorum-based CH (S-QCH) algorithms are proposed considering the asynchronous asymmetric scenario. Though, SUs are preassigned as sender or receiver in D-QCH before the data transmission, no role is preassigned to SUs in S-QCH. The degree of rendezvous of both S-QCH and D-QCH is $<N$ with large MTTR value $\approx 2N$ and $2N^2$, respectively. In [14], a fast rendezvous CH algorithm (FRCH) is proposed for the asynchronous environment, where the number of available channels N may be same or vary for the SUs. The length of one CH period in FRCH is $N(2N + 1)$, where N represents the number of rounds and $2N + 1$ signifies the number of time slots in each round. Besides, the MTTR value is equal to the length of one CH period, i.e., $N(2N + 1)$ in FRCH. Furthermore, the numbers of available channels are under-utilized in FRCH as the rendezvous occur at some particular available channels during the entire CH_Seq.

The enhanced alternate hop and wait (E-AHW) [15] protocol is designed for asynchronous symmetric and asymmetric environment. Each SU generates a 49-element ID sequence in E-AHW comprising "2" as the most significant element and the medium access control address of the SU. Each element of the ID sequence is associated with an elementary

CH_Seq. The degree of rendezvous in E-AHW is very small, i.e., $\leq 15\%$ with large MTTR value, i.e., $3nP$, where P is the smallest prime number $> N$ and n is the length of the element ID sequence. Liu *et al.* [16] have designed a jump-stay (JS) CH algorithm combing the jump and stay patterns in the asynchronous asymmetric environment. The CH_Seq is divided into $2N$ inner-rounds in JS, where each inner-round consists of a jump pattern and stay pattern of $2P$ and P number of slots, respectively, where $P > N$ and P is a prime number. According to the protocol, SUs hop on each available channel in the jump-pattern and stay on a specific channel in the stay-pattern. In JS, it is very difficult to achieve the rendezvous due to selection of different stay channels and random common available channel in the jump-pattern by the SUs. Moreover, the MTTR between two SUs is very large, i.e., $\approx 3P$.

An enhanced JS (EJS) CH algorithm is proposed in [17], where each round comprises $3P$ number of slot indexes for the jump pattern and P number of slot indexes for the stay pattern. However, the performance of EJS in terms of MTTR and degree of rendezvous is $\approx 4P$ and $\approx N$, respectively, which is not very impressive. The asynchronous CH prime sequences (ACHPSs) scheme is proposed in [18], where SUs can enter into the network without any global clock-synchronization. The CH_Seq of length p^2 is generated in ACHPSs by concatenating p number of periods, where each period is composed of p number of time slots and channels. The proposed protocol achieves higher rendezvous-success rate in two-user and multiuser cases by implementing different collision-avoidance schemes. However, the MTTR value of ACHPSs is very large, i.e., $p^2 - p$. Furthermore, each SU shares the same p number of available channels, which is not efficient in case of heterogeneous dynamic spectrum environment.

Dual non-ID and ID-based CH algorithms named as T-CH and D-CH are proposed in the asymmetric asynchronous environment considering prime number of channels [19]. Although T-CH tries to increase the degree of rendezvous, the collision-probability is also increased, specifically in multiuser environment. Besides, D-CH algorithm gives a larger MTTR value of $N^2 + 2N$ with less degree of rendezvous, if both SUs have different IDs. Monemi *et al.* [20] have analyzed the probable interference region generated by the SUs for the PUs in a CRN. However, the proposed work lacks sufficient analysis for the degree of rendezvous.

From the survey of the current literature, it is concluded that the performance of the most CH protocols in terms of degree of rendezvous, available channel utilization, and MTTR is not so encouraging. In this paper, an AACH protocol is proposed to increase the degree of rendezvous, minimize the MTTR and maximize the channel utilization. The main contributions of this paper can be summarized as follows.

1) The proposed AACH protocol can work for any number of channels.
2) AACH guarantees at least $2(|N| + 1)$ number of rendezvous between any two SUs even with single common available channel, where $|N|$ is the total number of available channels in the CRN.
3) AACH can achieve smaller MTTR value, which is $<2(|N| + 1)$.
4) The AACH protocol works efficiently in multiuser scenario without degrading the degree of rendezvous.

III. Problem Formulation

A. System Model

Consider a CRN that consists of a set of licensed PUs and K numbers of unlicensed SUs those reside in a common geographical region, where $K \geq 2$. Each SU is equipped with one half-duplex radio transceiver, which can sense the spectrum holes and performs the transmission of control or data messages. Let, N be the set of available licensed channels in the CRN indexed from [0, $|N|-1$] and the channels of set N are defined as $\{c_0, c_1, \ldots, c_j, \ldots, c_{|N|-1}\}$. Considering the conditions of asymmetric CH, let $C_M = \{c_0, c_2, c_j, \ldots, c_{|N|-2}\}$ and $C_N = \{c_2, c_3, c_j, \ldots, c_{|N|-1}\}$ be the different sets of available channels for SU_M and SU_N, respectively. For successful rendezvous in asymmetric scenario, there is at least one common channel between both SUs, which acts the operational or rendezvous channel. Hence, $\exists\ c_j \in (C_M \cap C_N)$ such that $C_M \cap C_N \neq \emptyset$.

The entire network is divided into multiple time slots or slot indexes those start from 0, 1,..., T_p-1, where T_p is length of the CH_Seq. The hopping process is executed among different SUs to find a common vacant channel in which negotiation and data transfer are accomplished during the communication. The CH pattern of a particular SU is determined by its CH_Seq and is expressed as CH_Seq = {(0, c_0), (1, c_1),..., (j, c_j),..., (T_p-1, c_{T_p-1})}, where $c_j \in [0, |N|-1]$ represents the channel number visited by an SU at jth time slot in a CH_Seq. Let, CH_Seq_M and CH_Seq_N be the CH sequences of SU_M and SU_N, respectively, with t_m and t_n as their entry slot indexes. Consider C_{MN} as the common channel set of both SUs. A channel $c_j \in C_{MN}$ is called as the rendezvous channel for both SUs in asynchronous environment, if $(c_j, t_r) = (c_j, t_s)$, but $(t_r \neq t_s)$.

B. Generation of Common Sequence Set

A distributed CH mechanism is proposed to generate the common sequence set (CS), which is used by the SUs to generate their CH Sequences. Algorithm 1 represents the generation of a matrix called SeqMat considering the concept of longest common subsequence (LCS) and *Addition Modular Arithmetic*. The designed SeqMat consists of $(|N|+1)$ number of rows and $(|N|+1)$ number of columns, where each row and column represents individual *Common Sequences*. According to the concept of LCS, the initial row and column is occupied by empty sequences, i.e., either 0 or ϕ. Let, a_{ij} be any element of LCS matrix. The value of a_{ij} for first row and column can be defined as $LCS_{i\times j} = a_{ij} = 0$, when $i = j = 0$. In the proposed algorithm, the number 0 is replaced with common available channel C_c of both the SUs. The reason to place common available channel in the row_0 and col_0 is to increase the degree of rendezvous. For the generation of rest of the elements of SeqMat, the *Addition Modular Arithmetic* is used.

According to Algorithm 1, *Addition Modular Arithmetic* is applied across individual rows except row_0, which is already occupied by the common available channel C_c. Each row

Algorithm 1 Generation of Common Sequence Matrix *SeqMat*

Input: N: Set of available channels in the CRN.

Output: $SeqMat[i][j]$: Common Sequence Matrix.

Notations:

AV_{SU_i}: Available channels set for any SU i; AV_{SU_j}: Available channels set for any SU j; CAV_{SU_i,SU_j}: Empty set which will store the common available channels for any SU i and j.

1: Initialize $row = col = 0$, i = j = 0;
2: $CAV_{SU_i,SU_j} = AV_{SU_i} \bigcap AV_{SU_j}$;
3: $C_c = rand[CAV_{SU_i,SU_j}]$;
4: **if** ($row == 0$ && $col == 0$) **then**
5: **for** $row = 0$ **to** $|N|$ **do**
6: $SeqMat[row][col] = C_c$;
7: **end for**
8: $row = 0$;
9: **for** $col = 0$ **to** $|N|$ **do**
10: $SeqMat[row][col] = C_c$;
11: **end for**
12: **end if**
13: **for** $row = 1$ **to** $|N|$ **do**
14: $j = 0$;
15: **for** $col = 1$ **to** $|N|$ **do**
16: $SeqMat[row][col] = (N[i] + N[j])\%|N|$;
17: $j = j + 1$;
18: **end for**
19: $i = i + 1$;
20: **end for**
21: Return *SeqMat*

starting from row_1 is associated with individual channel of set N, i.e., [0, $|N|-1$]. In other words, row_1 is associated with channel 0, row_2 with channel 1, and so on. To calculate the elements of individual row, its corresponding channel is going to be added with the channels of set N. For example, the initial channel, i.e., channel 0 is added with each channel of set N across all the columns starting from col_1 to $\text{col}_{|N|}$ to generate the elements of row_0. The elements placed in row_1 are generalized as $\text{SeqMat}[\text{row}_1][\text{col}_1] = \{(N[0] + N[0])\}\%|N|$, $\text{SeqMat}[\text{row}_1][\text{col}_2] = \{(N[0] + N[1])\}\%|N|, \ldots, \text{SeqMat}[\text{row}_1][\text{col}_{|N|}] = \{(N[0] + N[|N| - 1])\}\%|N|$. Thus, the rest of the elements of the SeqMat is calculated.

Each row and column of the matrix SeqMat is associated with individual sequence known as *Common Sequence* and each element represents specific channel number. Hence, $2(|N|+1)$ numbers of common sequences are generated from $|N|$ number of channels out of which $s_0, \ldots, s_{|N|}$ are obtained by considering the sequences across all the rows from left to right and $s_{|N|+1}, \ldots, s_{2(|N|+1)}$ are occurred by considering bottom up approach across all the columns. Hence, channels of row_0 are associated with common sequence s_0. Likewise, col_0 is associated with the common sequence $s_{|N|+1}$, and so on. Finally, the CS is generated by concatenating individual common sequence of SeqMat as given in Algorithm 2. Upon generating the common sequences, each SU has to choose the

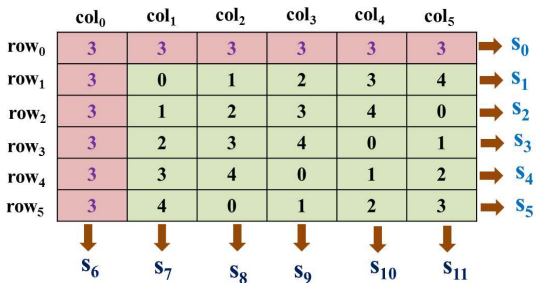


Fig. 1. Generation of common sequence matrix SeqMat.

permutation of any $|N| + 1$ number of common sequences out of $2(|N| + 1)$ number of common sequences to generate its CH_Seq.

Let us consider an example, where $|N| = 5$, i.e., $\{0, 1, 2, 3, 4\}$. Let, $\text{SU1}_{AV} = \{0, 1, 3\}$ and $\text{SU2}_{AV} = \{2, 3, 4\}$ be the available channel sets for SU1 and SU2, respectively. Hence, the common available channel between both SUs is $\{3\}$. Based on Algorithm 1, a SeqMat matrix of $(|N|+1) \times (|N|+1)$, i.e., 6×6 is created as shown in Fig. 1. According to Algorithm 1, both initial row (row_0) and column (col_0) are occupied by common available channel, i.e., 3. The rest of the values starting from row_1 to $\text{row}_{|N|}$, such as row_1 to row_5 of SeqMat are generated by considering *Addition Modular Arithmetic*, which is given in Algorithm 1. The values of row_1 beginning from col_1 to $\text{col}_{|N|}$ is obtained by adding the initial channel 0 with each channel of the set N starting from 0 to 4. The values of row_1 in SeqMat are $\text{SeqMat}[\text{row}_1][\text{col}_1] = (0 + 0)\%5 = 0$, $\text{SeqMat}[\text{row}_1][\text{col}_2] = (0 + 1)\%5 = 1$. Likewise, $\text{SeqMat}[\text{row}_1][\text{col}_{|N|}] = (0 + 4)\%5 = 4$. Hence, the values of row_1 become $<3, 0, 1, 2, 3, 4>$, where the initial channel 3 represents the common available channel. Upon proceeding this way, the final row row_5 is obtained as $\text{SeqMat}[\text{row}_5][\text{col}_1] = (4 + 0)\%5 = 4$, and goes on up to $\text{SeqMat}[\text{row}_5][\text{col}_5] = (4 + 4)\%5 = 3$, as shown in Fig. 1. It is to be noted that each row and column of SeqMat is associated with individual common sequence. Hence, based on Algorithms 1 and 2, from $|N| = 5$, $2(|N|+1)$, i.e., 12 numbers of common sequences are generated, such as $s_0, s_1, \ldots, s_{11}$.

Common sequences $s_0, s_1, \ldots, s_5$ are generated considering all rows from row_0 to row_5 known as the row related common sequences. Similarly, $s_6, \ldots, s_{11}$ are obtained from all columnar values starting from col_0 to col_5 in bottom up approach and is known as the column related common sequences. All common sequences starting from $s_0, \ldots, s_{11}$ are shown with arrow ($\rightarrow$) mark in Fig. 1. For generation of CH_Seq, each SU selects a permutation of $(|N|+1)$, i.e., 6 numbers of common sequences out of total 12 numbers of generated common sequences.

Lemma 1: An SU visits all the channels of set N, i.e., [0, $|N| - 1$] in each common sequence within $N + 1$ slot indexes.

Proof: It is already explained that the initial or the final slot index of a common sequence is occupied by the common available channel C_c, where rest of the channels are obtained by *Addition Modular Arithmetic*. Let us consider a common

Algorithm 2 Generation of CS

Input: *SeqMat*[*i*][*j*]: Sequence Matrix.
Output: *CS*: Set of Common Sequences.
Notations: *s*: Temporary Empty set;

```
Initialize row = col = 0, index = j = 0, CS = s = φ;
for row = 0 to |N| do
    for col = 0 to |N| do
        s[index] = SeqMat[row][col];
        index = index + 1;
    end for
    CS =< CS || s >;
end for
for col = 0 to |N| do
    for row = |N| to 0 do
        s[index] = SeqMat[row][col];
        index = index + 1;
    end for
    CS =< CS || s >;
end for
Return CS
```

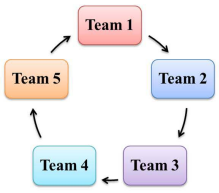
sequence s_i that contains a particular channel c_r twice without counting its presence in the initial or final slot index. The above statement can be generalized as follows. $N[i]+c_d = c_r$ and $N[i]+c_{d'} = c_r$, which implies $c_d = c_{d'}$. From the set N, it is observed that the difference between any two channel is 1, i.e., $c_i - c_j = 1$, where $i > j$. Similarly, the difference between c_d and $c_{d'}$ can be calculated as $c_d - c_{d'} = 1$, which contradicts that $c_d = c_{d'}$. Hence, the lemma follows. ■

Lemma 2: The number of common channel overlapping between any two common sequences can be $0 \leq 1 \leq (|N|+1)$.

Proof: The common channel overlapping takes place, when both common sequences contain the common channel at the same slot index. According to the proposed protocol, there must be a complete common channel overlapping, i.e., $(|N| + 1)$ between any combination of s_0 and $s_{|N|+1}$. However, there will be no overlapping between any two common sequences s_i and s_j, where $s_i \in \{s_0, \ldots, s_{|N|}\}$, $s_j \in \{s_{|N|+1}, \ldots, s_{2(|N|+1)}\}$ and *vice versa*. Besides, the number of channel overlapping between any two common sequences could be 1, if both of them belong to row related sequences $\{s_0, \ldots, s_{|N|}\}$ or column related sequences $\{s_{|N|+1}, \ldots, s_{2(|N|+1)}\}$. Furthermore, an SU can select $N+1$ number of common sequences out of $2(|N|+1)$ number of common sequences in $P\{2(|N|+1), |N|+1\} = x$ number of ways, whose value is $>>> |N|$. Hence, there is higher chance of common channel overlapping among any two CH sequences due to occurrence of common available channel either at the initial or final index of the common sequences. ■

IV. Asymmetric Asynchronous Channel Hopping Mechanism

To achieve rendezvous in asymmetric asynchronous environment is very challenging as SUs enter into the network at different instances of time with different sets of available channels. For establishing the communication among both SUs in asynchronous scenario, the slot indexes should overlap with each other in the common available channel C_c. In the proposed AACH protocol, the concept of round Robin tournament (RRT) [21] is used for achieving the guaranteed rendezvous.



(a)

Rounds \ Teams	1	2	3	4	5
1	5	4	6 (bye)	2	1
2	6 (bye)	5	4	3	2
3	2	1	5	6 (bye)	3
4	3	6 (bye)	1	5	4
5	4	3	2	1	6 (bye)

(b)

Fig. 2. Example of RRT.

A. Round Robin Tournament

Let us consider an RRT that consists of N number of teams. In order to construct the round Robin schedule for N number of teams, let us take an example for $N = 5$. Since, N is odd here, let a dummy team (say team number 6) be added for the fair distribution of the teams in each round. Let, all $N = 5$ number of teams be arranged in a circular manner as shown in Fig. 2(a). The teams are placed along the row and rounds are presented along the column of a matrix as shown in Fig. 2(b). Based on the rules given in [21], the first "bye" is placed in the column number $([N+1]/2)$, i.e., $([5+1]/2) = 3$ in Round 1. Hence, Team 3 is going to play with dummy Team 6 in Round 1 and draw a bye. In order to calculate the team in each successive round that needs to play with the dummy Team 6, a clockwise movement of $([N+1]/2)$ number of steps is performed starting from the team that draw a bye in the previous round. Applying the same rule for Round 2, Team 1 will play with Team 6 and draw a bye after the clockwise movement of 3 number of steps starting from Team 3. Similarly, in Round 3, Round 4, and Round 5, the teams, such as Team 4, Team 2, and Team 5, respectively, will pair with Team 6 as shown in Fig. 2(b).

In the first round, the teams are placed starting from N through 1 in each empty cells to get the pairing, such as N, $N-1, \ldots,$ bye, $\ldots, 2, 1$ and by skipping the cell that is already occupied by bye. As shown in Fig. 2(b), the ordering of teams that occupies the cells in the Round 1 is Team 5, Team 4, bye, Team 2 and Team 1. Similarly, the teams in the next round are obtained by moving one place along the clockwise direction from the teams of the previous round. For example, one place right from the Team 5 is Team 1, which should be allocated in the first cell of Round 2. Since, it is already occupied with bye, this cell needs to be skipped. Next to the bye in Round 2, it is the Team 5 located one step right to the Team 4 of the previous round. Thus, the playing schedule for other teams can be calculated as shown in Fig. 2(b), where a bye is appeared in each round after addition of the dummy Team 6.

B. Creation of Round Robin Matrices

Let us consider the whole round Robin schedule as a matrix consisting of N number of rows and columns, where the

values inside the matrix are individual channel numbers. The presence of bye in each row can be remapped with certain channels. The main idea is to replace the common available channel in the slot indexes as well as channel numbers of a common sequence whose values are identified by the remapped channels. Suppose, bye is remapped with channel j, then the slot index and channel number j of each common sequence are replaced by the common available channel C_c.

As given in lines 2–4 of Algorithm 3, the set of total available channels N is divided into three groups, such as $N_{\text{leading}}(N_l)$, $N_{\text{central}}(N_c)$, and $N_{\text{following}}(N_f)$. The reason behind this division is to incorporate the RRT concept in N_l and N_f, which increases the utilization of common channels inside a common sequence. It is previously discussed that a bye can only occur if the total number of channel is odd. Hence, upon finding $|N_f|$ or $|N_l|$ is even, the channel of N_c is combined with the set having even cardinality to make it odd, which is explained in lines 5–10 of Algorithm 3. Let us consider the channels present inside the set $N_l = \{C_a, C_b, \ldots, C_d\}$ and $|N_l|$ is even. In order to make $|N_l|$ as odd, a union operation $N_l \cup N_c = \{C_a, C_b, \ldots, C_d, C_e\}$ is carried out, where $C_e \in N_c$.

In Algorithm 3 from lines 11–42, the procedures of creation of round Robin matrices, such as RRMat_l and RRMat_f are described. The total channels of set N_l divided into a matrix of 0 to $|N_l|$ number of rows and columns is known as RRMat_l. For common channel replacement, only the remapped channels are needed that appeared in the place of bye. Lines 11–25 represent the remapping of channels in the corresponding row and column in the similar place with occurrence of bye. In the proposed protocol, the word bye is used as place of reference for the placement of the remapped channels. However, all calculation and common available channel replacement is based on the number of remapped channels. Let $|N_l|$ be the number of channels present in the set N_l. In the first row, the bye is placed in the column number equal to value $= ([|N_l| + 2]/2)$, which is given in lines 12–15. For the placement of rest of the bye in the remaining rows, the movement should be along clockwise direction by $([|N_l| + 2]/2)$ number of steps starting from the previous generated value and should continue until a bye is placed in every row as given in lines 16–19. As shown in lines 21 and 22, the remapped channels are placed in the same row and column with bye in RRMat_l, where the index of remapped channel is equal to the value-th element of set N_l, i.e., N_l[value]. The calculated channel number is used for the common channel replacement in each common sequence. The same procedure is also applied to the set N_f, where the total schedule is divided into 0 to $|N_f|$ number of rows and columns.

Algorithm 3 Creation of Round Robin Matrices

Input: N: Set of licensed channels in the CRN.

Output: $RRMat_l$[][]: Round Robin Matrix generated from the channels of set N_l; $RRMat_f$[][]: Round Robin Matrix generated from the channels of set N_f.

Notations:

N_l: Set that stores front segment of channels of set N;
N_f: Set that stores back segment of channels of set N;
N_c: Set that stores the middle channel channel of set N;
$replace_l$[]: Stores the values of remapped channels generated; from the matrix $RRMat_l$; $replace_f$[]: Stores the values of remapped channels generated from the matrix $RRMat_f$.

1: Initialize $N_l = N_c = N_f = \phi$;
2: $N_c = |N|/2$;
3: $N_l = \{0, 1, \ \ldots, \ (|N|/2) - 1\}$;
4: $N_f = \{(|N|/2) + 1, \ \ldots, \ |N| - 1\}$;
5: **if** $(|N_l|\%2 == 0)$ **then**
6: $N_l = N_l \cup N_c$;
7: **end if**
8: **if** $(|N_f|\%2 == 0)$ **then**
9: $N_f = N_f \cup N_c$;
10: **end if**
11: **for** $row = 0$ **to** $|N_l|$ **do**
12: **if** $(row == 0)$ **then**
13: $value = (\frac{|N_l|+2}{2}) - 1$;
14: $previous\ value = value$;
15: $col = value$;
16: **else**
17: $value = \{previous\ value + (\frac{|N_l|+2}{2})\}\%N_l$;
18: $previous\ value = value$;
19: $col = value$;
20: **end if**
21: $RRMat_l[row][col]$ = "bye";
22: $RRMat_l[row][col] = N_l[value]$;
23: $replace_l[i] = RRMat_l[row][col]$;
24: $i = i + 1$;
25: **end for**
26: $i = 0$;
27: **for** $row = 0$ **to** $|N_f|$ **do**
28: **if** $(row == 0)$ **then**
29: $value = (\frac{|N_f|+2}{2}) - 1$;
30: $previous\ value = value$;
31: $col = value$;
32: **else**
33: $value = \{previous\ value + (\frac{|N_f|+2}{2})\}\%N_f$;
34: $previous\ value = value$;
35: $col = value$;
36: **end if**
37: $RRMat_f[row][col]$ = "bye";
38: $RRMat_f[row][col] = N_f[value]$;
39: $replace_f[i] = RRMat_f[row][col]$;
40: $i = i + 1$;
41: **end for**
42: Return $RRMat_l$ *and* $RRMat_f$;

C. Generation of CH Sequences

As given in Algorithm 4, the remapped channels are selected from those row and column numbers, whose values are same as the common available channel number C_c. Sometimes, the index of common channel $C_c \geq |\text{RRMat}_l|$ or $|\text{RRMat}_f|$. In this case, a modular operation is carried out as C_c % $|\text{RRMat}_l|$ or $|\text{RRMat}_f|$. If the SUs have more than one common available channels, more than one row and column number are selected for the common channel replacement. In Algorithm 4, the remapped channel number is calculated by considering both matrices RRMat_l and RRMat_f. After calculation of remapped channels based on the individual common available channel, each SU executes Algorithm 5 for generating its CH_Seq.

Algorithm 4 Selection of Remapped Channel Number From Round Robin Matrices

Input: N_l; N_f; $RRMat_l$; $RRMat_f$; $replace_l[]$; $replace_f[]$; C_c: Selected common available channel between SU1 and SU2;

Output: ch_{r_l} : Remapped channel generated from the row considering set N_l; ch_{c_l} : Remapped channel generated from the column considering set N_l; ch_{r_f}: Remapped channel generated from the row considering set N_f; ch_{c_f} : Remapped channel generated from the column considering set N_f; CAV_{SU_i,SU_j}: Set of common available channels.

Notations:

C_{ctotal}: Total number of available channels for both the SUs.

1: $C_{ctotal} = |CAV_{SU_i,SU_j}|$;
2: **for** $i = 1$ **to** C_{ctotal} **do**
3: **if** $C_c \geq |RRMat_l|$ *or* $|RRMat_f|$ **then**
4: $num = C_c\%(|RRMat_l|$ *or* $|RRMat_f|)$;
5: **end if**
6: **for** $col = 0$ **to** $|N_l| - 1$ **do**
7: **for** $index = 0$ **to** $sizeof(replace_l)$ **do**
8: **if** $RRMat_l[num][col] == replace_l[index]$ **then**
9: $ch_{r_l} = replace_l[index]$;
10: **end if**
11: **end for**
12: **end for**
13: **for** $row = 0$ **to** $|N_l| - 1$ **do**
14: **for** $index = 0$ **to** $sizeof(replace_l)$ **do**
15: **if** $RRMat_l[row][num] == replace_l[index]$ **then**
16: $ch_{c_l} = replace_l[index]$;
17: **end if**
18: **end for**
19: **end for**
20: **for** $col = 0$ **to** $|N_f| - 1$ **do**
21: **for** $index = 0$ **to** $sizeof(replace_f)$ **do**
22: **if** $RRMat_f[num][col] == replace_f[index]$ **then**
23: $ch_{c_f} = replace_f[index]$;
24: **end if**
25: **end for**
26: **end for**
27: **for** $row = 0$ **to** $|N_f| - 1$ **do**
28: **for** $index = 0$ **to** $sizeof(replace_f)$ **do**
29: **if** $RRMat_f[row][num] == replace_f[index]$ **then**
30: $ch_{c_f} = replace_f[index]$;
31: **end if**
32: **end for**
33: **end for**
34: Return $ch_{r_l}, ch_{c_l}, ch_{r_f}, ch_{c_f}$;
35: Execute Algorithm 5 for creation of CH sequence considering the obtained remapped channels;
36: **end for**

The generation of CH_Seq using those remapped channels from Algorithm 4 is deduced in Algorithm 5. Each SU selects a permutation of $|N|+1$ number of common sequences before generating its CH_Seq. The selected remapped channel numbers signify the channel numbers and slot indexes, where the common available channel C_c is going to be replaced inside the selected common sequence s_i. Let, c_f and c_l be the remapped channels, which are obtained from the round Robin matrices. Then the common channel C_c is going to be replaced in the slot indexes along with channel numbers equal to c_l and c_f inside the common sequence s_i. As given in line 13 of Algorithm 5, the CH sequences are generated by each SU considering the modified common sequences.

Algorithm 5 Generation of CH_Seq in AACH

Input: *CS*: *Common Sequence* Set.
CAV_{SU_i,SU_j}: Set of common available channels.
ch_{r_l}; ch_{c_l}; ch_{r_f}; ch_{c_f}; C_c.

Output: *CH_seq*: Channel Hopping Sequence.

Notations: *s*: Individual *Common Sequence*.

1: Initialize $seq\_num = 0$, $CH\_seq = \phi$;
2: Each SU choose $|N| + 1$ number of common sequences randomly from set *CS*;
3: **while** $seq\_num < |N| + 1$ **do**
4: Process the selected individual common sequences for common channel replacement;
5: **for** $index = 0$ **to** $|N|$ **do**
6: **if** $(s[index] == ch_{r_l}||s[index] == ch_{c_l}||s[index] == ch_{r_f}||s[index] == ch_{c_f})$ **then**
7: $s[index] = C_c$;
8: **end if**
9: **if** $(index == ch_{r_l}||index == ch_{c_l}||index == ch_{r_f}||index == ch_{c_f})$ **then**
10: $s[index] = C_c$;
11: **end if**
12: **end for**
13: $CH\_seq =< CH\_seq \parallel s >$;
14: $i = i + 1$;
15: $seq\_num = seq\_num + 1$;
16: **end while**

Let us consider an example, where the total number of available channels is $|N| = 5$, which are indexed as $\{0, 1, 2, 3, 4\}$. Let the available channels for SU1 and SU2 be $SU1_{AV} = \{0, 3, 4\}$ and $SU2_{AV} = \{1, 3, 5\}$, respectively. Hence, the number of common available channels between SU1 and SU2 is 3. Let the entry slot index of SU1 and SU2 be 0 and 1, respectively. Based on Algorithm 3, the channel present in the set $N_c = \{|N|/2\} = \{5/2\} = \{2\}$. Similarly, the channels present in the set $N_l = \{0, 1, \ldots, (|N|/2 - 1)\} = \{0, 1\}$ and $N_b = \{(|N|/2 + 1), \ldots, (|N| - 1)\} = \{3, 4\}$. As both $|N_l|$ and $|N_f|$ are even, the channels of set N_c need to be added with the channels of both sets N_l and N_f to generate bye in each column and row. After addition of the channel $2 \in N_c$, the updated sets of N_l and N_f are $\{0, 1, 2\}$ and $\{2, 3, 4\}$, respectively, as shown in Fig. 3(a). To calculate the remapped channel, let the channels of set N_l and N_f be arranged in a ring as given in Fig. 3(a). By considering the number of channels of set N_l and N_f, the round Robin matrices such as RRMat_l and RRMat_f are generated as shown in Fig. 3(b).

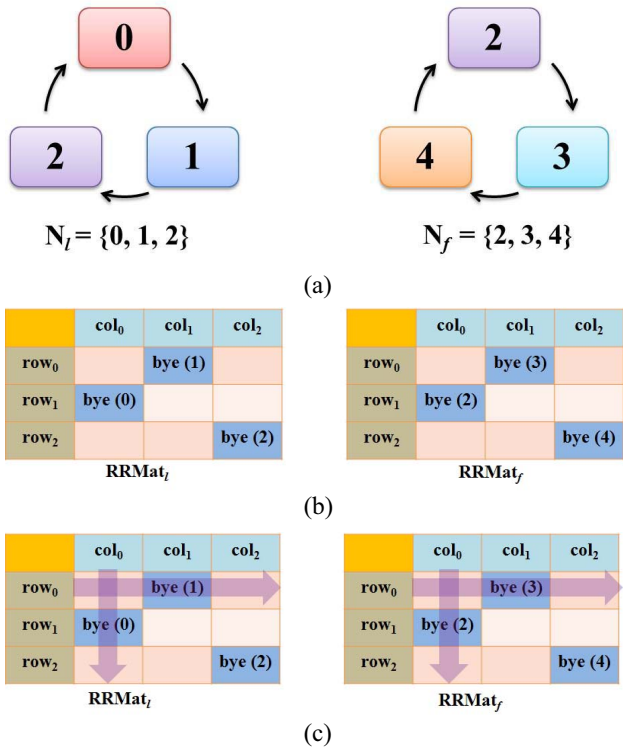


Fig. 3. Selection of remapped channels using RRT concept. (a) Representation of channels in set N_l and N_f. (b) RRMat$_l$ and RRMat$_f$. (c) Selection of remapped channels.

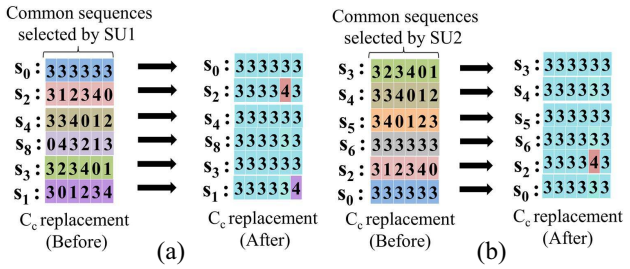


Fig. 4. Replacement of C_c in common sequences having remapped channels 0, 1, 2, and 3.

Now, in the matrix RRMat$_l$, the first bye is placed in row 0 and the column number is calculated as value $= ([|N_l| + 2]/2) - 1 = ([|3| + 2]/2) - 1 = 1$. For easy understanding, bye is only used as the place of reference for the remapped channels. Hence, the first bye is placed in column 1 in row$_0$. Then, the bye is remapped with the value-th element of set N_l, i.e., $N_l[\text{value}] = N_l[1] = 1$. Similarly, for row$_2$ of matrix RRMat$_l$, the bye is allocated in the column number value $= \{\text{previous value} + ([|N_l| + 2]/2)\}\%|N_l| = \{1 + ([|3| + 2]/2)\}\%|3| = 0$. Hence, in row$_2$, bye is placed in col$_0$ and is remapped with the channel number of set N_l, i.e., $N_l[0] = 0$. Similarly, the bye and the remapped channels are placed in the rest of the rows and columns of both matrices RRMat$_l$ and RRMat$_f$, which is shown in Fig. 3(b).

Certain remapped channels are selected from both RRMat$_l$ and RRMat$_f$, where the column and row number is equal to the common channel number (C_c) of both SUs. The remapped channels represent the channels and slot indexes, where the common channel is going to be replaced inside the common sequence. In the discussed example, the common available channel $C_c = |\text{RRMat}_l| = |\text{RRMat}_f| = 3$. Hence, a modulation operation is carried out as $3\%\ (|\text{RRMat}_l|) = 0$ and $3\%\ (|\text{RRMat}_f|) = 0$. Here, the value 0 signifies the row$_0$ and col$_0$ of both the matrices RRMat$_l$ and RRMat$_f$. Upon executing the modulation operation, the remapped channels are selected from the row$_0$ and col$_0$ of RRMat$_l$, i.e., channels 0 and 1 along with channels 2 and 3 from RRMat$_f$ as shown in Fig. 3(c).

In the next step, each SU selects $|N| + 1$ number of common sequences from the CS. The set of common sequences generated from $|N| = 5$ are CS $= \{s_0, s_1, s_2, \ldots, s_{11}\}$, those are shared by both SU1 and SU2. Let, $<s_0, s_2, s_4, s_8, s_3, s_1>$ and $<s_3, s_4, s_5, s_6, s_2, s_0>$ be the permuted common sequences selected by SU1 and SU2, respectively. Now, based on Algorithm 5, the common available channel $C_c = 3$ is going to be replaced in the channel numbers and slot indexes 0, 1, 2, and 3 of each common sequence for both SUs. The channel order of selected common sequences before and after the replacement of C_c is shown in Fig. 4. Upon replacing the common available channel, each SU creates its CH_Seq by concatenating the modified common sequences. Fig. 5 shows the rendezvous between both SUs in the common available channel number 3 after applying the concept of RRT for which the degree of rendezvous is 33 out of 36 number of slot indexes.

V. Analysis of CRN Metrics

In this section, we would like to analyze different metrics of the CRN based on the designed AACH algorithm those influence the performance of the IoT devices.

A. Degree of Rendezvous

Degree of rendezvous is defined as the number of times two SUs meet with each other in a CH_Seq. In AACH, the degree of rendezvous is deduced by considering different scenarios explained in the Lemma 3.

Lemma 3: In AACH protocol, degree of rendezvous between any two SUs is $\leq (\text{CH}_{\text{length}} - \beta_{\text{total}})$, where $\text{CH}_{\text{length}}$ is the length of the CH_Seq, β is the total number of slot indexes, where the common channel is not replaced in a CH_Seq.

Proof: The β_{total} in a CH_Seq is defined as the summation of the slot indexes having no common channel replacement. Hence, $\beta_{\text{total}} = \sum_{i=0}^{|N|} \beta_{cs_{i(\text{SU1})}} + \sum_{j=0}^{|N|} \beta_{cs_{j(\text{SU2})}}$, where $\beta_{cs_{i(\text{SU1})}}$ and $\beta_{cs_{j(\text{SU2})}}$ represent the β value in each common sequence selected by both SU1 and SU2, respectively. Furthermore, it is observed that the value β varies with the number of common available channels and the selection of remapped channels from RRMat$_l$ and RRMat$_f$. The individual value of β in each common sequence can be calculated by considering different common channel replacement scenarios.

Scenario 1: If the remapped channel is selected from the last row and column of RRMat$_l$ and RRMat$_f$ and the common available channel is not equal to the remapped channel.

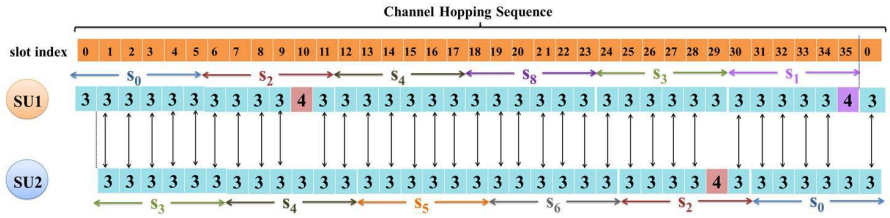


Fig. 5. Rendezvous between two SUs in AACH.

In this scenario, the number of remapped channel in a common sequence is at most $8 \times C_{\text{ctotal}}$, where C_{ctotal} is the number of common available channels. Each row and column of RRMat_l and RRMat_f gives unique remapped channels. Individual β value in a common sequence falls within the range $0 \leq \beta \leq ([|N|+1]/2)$ for $|N| < 8$ and $0 \leq \beta \leq (|N|+1)-8$ for $|N| > 8$ as the maximum number of common available channel replacement in a common sequence is 8.

Let us consider an example as explained in Fig. 5, where the remapped channels are 0, 1, 2, and 3. It can be observed that the rendezvous occur in every slot indexes and channels of common sequences that are already replaced by common available channel 3 either due to the remapping or default presence of common available channel. However, the number of slot indexes, where the common available channel is not replaced could be $\beta_{\text{total}} = \beta_{s_{2(\text{SU1})}} + \beta_{s_{1(\text{SU1})}} + \beta_{s_{2(\text{SU2})}} = 1 + 1 + 1 = 3$, which is equal to the number of nonrendezvous slot indexes of both SU1 and SU2. As shown in Fig. 5, the total number of rendezvous = $\{CH_{\text{length}} - \beta_{\text{total}}\} = 36 - 3 = 33$ out of 36 slot indexes.

Scenario 2: If the remapped channel is selected from last row and column of RRMat_l and RRMat_f, where the common available channel is equal to any of the remapped channels.

In this scenario, the number of remapped channel in each common sequence is equal to 1 as the intersection of last row and column generates a single remapped channel from both of the matrices. Hence, the number of replacement is ≤ 4 in a common sequence including both channel and slot index replacement. Here, β is defined as $0 \leq \beta < (|N|+1)-4$. Let us consider an example, where $C_c = 2$. Based on the value of C_c, the remapped channels, such as 2 and 4 are selected from row_2 and col_2 of both the matrices RRMat_l or RRMat_f. The degree of rendezvous in Scenario 2 is less than Scenario 1 as the number of remapped channel in Scenario 2 is only two, i.e., 2 and 4. Hence, the number of common channel replacement in each common sequences is $< = 4$.

After analyzing the individual degree of rendezvous in both scenarios, the degree of rendezvous in AACH can be generalized as $\{0 \leq \text{degree of rendezvous} \leq (|N|+1)^2 - \beta\}$, i.e., $\{0 \leq \text{degree of rendezvous} \leq CH_{\text{length}} - \beta\}$, where β value lies in between $0 \leq \beta < |N|+1$. ■

B. Maximum Time to Rendezvous

The MTTR is defined as the maximum number of slot indexes taken by an SU for the initial rendezvous with another SU in a CH_Seq, which is considered as the worst case of the TTR. In AACH, a generalized MTTR value can be deduced as given in Lemma 4.

Lemma 4: In AACH, the MTTR is $<2(|N|+1)$, where $(|N|+1)$ is length of the common sequence.

Proof: In AACH, if the number of available channel $C_{\text{ctotal}} > 2$, the MTTR is $<|N|+1$. The worst case of TTR arises when the remapped channel is selected from the last row and column of RRMat_l and RRMat_f and the single common available channel is equal to any of the remaped channels. Let us consider a scenario, where both SUs are having a common available channel $C_c = 2$, where $C_c \in C_{\text{ctotal}}$. Thus, the number of remapped channels selected from RRMat_l and RRMat_f are only two, i.e., 2 and 4 as the common available channel 2 represents the last row and column index of both the matrices. Furthermore, from Fig. 6. It is observed that both SUs wait more than $|N|+1$ number of slot indexes for rendezvous having single common available channels $C_c = 2$. Hence, it is concluded that the MTTR value is affected by the permutation order of common sequences as well as the entry slot index of the SUs. In order to achieve rendezvous within a short interval of time, a novel mechanism known as slot sacrifice (SS) is adopted in the designed AACH protocol.

The SS mechanism is implemented by the later entry SU after finding no rendezvous within $|N|+1$ number of slot indexes. Based on the example shown in Fig. 6, the SU2 will check how many number of slot indexes is behind of SU1. Let it be θ numbers of slot indexes. Let the time duration of each slot index be t_t. In case of finding no rendezvous in the first common sequence, SU2 will halt its hopping for a certain interval of time. Let it be t_h for θ numbers of slot indexes, where $t_h = \theta \times t_t$. After the expiry of t_h duration, SU2 resumes its hopping from the second common sequence. However, at the end of the CH_Seq, SU2 will sacrifice θ number of slot indexes to maintain the uniformity of the hopping duration with SU1. Although, SU2 will sacrifice θ number of slot indexes at the end, it will gain at least $|N|+1$ number of rendezvous in the previous slot indexes.

Let us consider an example, as shown in Fig. 7 to explain the SS mechanism. To show the significant improvement in the waiting time, the same CH scenario is considered as shown in Fig. 6. In Fig. 7, $\theta = 1$ as SU2 is one slot index behind of the SU1 and does not have any rendezvous in the first common sequence s_9. Hence, SU2 adopts SS mechanism to have rendezvous, where it will halt its hopping for the time duration of $t_h = 1 \times t_t$. This halting duration is equal to the duration of the slot index = 7 and continues its transmission from the slot index = 8 starting from the second common sequence. After waiting for one slot index duration, SU2 can

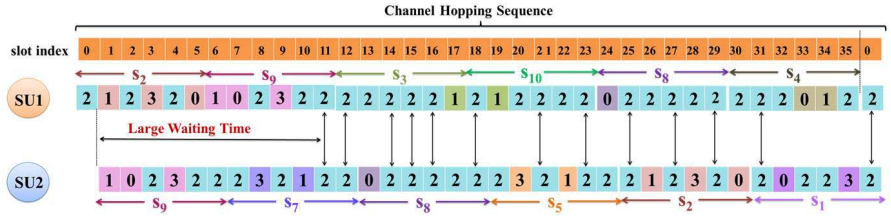


Fig. 6. Example of longer waiting time for rendezvous in AACH.

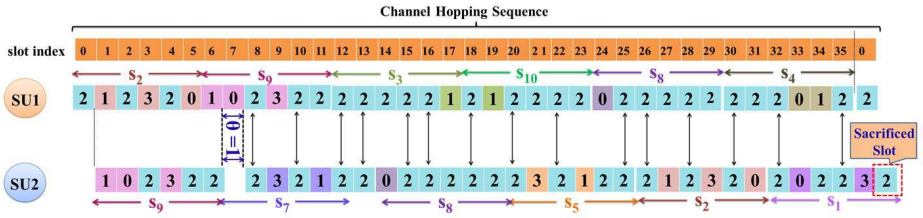


Fig. 7. Rendezvous between two SUs in AACH after adopting SS mechanism.

achieve the rendezvous immediately in the slot index = 8, which is the beauty of the purposed SS mechanism. Upon adopting the SS mechanism, AACH guarantees rendezvous within $2(|N|+1)$ number of slot indexes regardless the permutation type or entry slot indexes of SUs. Hence, the MTTR of AACH is $<2(|N|+1)$. ■

Lemma 5: The AACH protocol guarantees at least $2(|N|+1)$ number of rendezvous in an asymmetric asynchronous scenario.

Proof: By analyzing Lemmas 3 and 4, it is concluded that the purposed AACH mechanism grantees at least $2(|N|+1)$ number of rendezvous in a CH sequences even if with single common available channel and irrespective of the permutation or entry slot indexes of SUs. ■

C. Average Time to Rendezvous

Average TTR (ATTR) is calculated by considering all possible values of TTRs including the minimum and maximum values of TTR in the system. Hence, average of all possible values of TTRs is taken into account to calculate the ATTR.

Lemma 6: In AACH, the ATTR is $<(|N|+1)$, where $|N|$ is total number of available channels in the network.

Proof: In Lemma 4, it is proved that the minimum value of the TTR in AACH is 0, whereas the maximum value of the TTR is $<2(|N|+1)$. Hence, ATTR can be calculated by considering all possible values of TTRs in the network. Accordingly, ATTR in AACH = $([\sum(\text{All possible values of TTRs})]/[\text{Total numbers of TTRs }]) = ([0+1+\cdots+2(|N|+1)]/[2(|N|+1)]) < (|N|+1)$, since MTTR $< 2(|N|+1)$. ■

D. Maximum Inter Rendezvous Interval

The number of slot indexes between any two consecutive rendezvous in a CH_Seq is defined as the inter rendezvous interval (IRI). The maximum value of all possible IRIs in that CH_Seq is called as maximum IRI (MIRI). If the first rendezvous between two SUs occurs at slot t_i and next one occurs at t_j, IRI between the SUs is $\{(t_j - t_i) - 1\}$ and MIRI is the $\text{MAX}\{(t_j - t_i) - 1\}$. In a CRN, the MIRI should be minimized to reduce the waiting period between the first and the next rendezvous.

Lemma 7: The MIRI of AACH is $|N|+1$.

Proof: The MIRI of AACH is $|N|+1$ in case of single common available channel, which is one of the remapped channel as shown in Fig. 6. However, with increase in number of common available channel, the IRI can be minimized. ■

VI. Performance Evaluation

The performance evaluation of the proposed AACH protocol is performed using OMNeT++ simulator IDE version 4.5 considering the asymmetric channel model. It is considered that each SU can have different sets of available channels with at least one common available channel among them and can enter to the network at different slot indexes fulfilling the asynchronous condition. Performance of AACH is compared with similar protocols, such as SARCH [3], FRCH [14], JS [16], and EJS [17]. The proposed asymmetric simulation environment is implemented with 100 numbers of randomly deployed SUs with different numbers of available channels in presence of 50 numbers of licensed PUs over an area of 1000 m × 1000 m. The simulated CRN is divided into multiple slot indexes, where the duration of each slot index is considered to be 0.02 s.

In the designed simulation environment, the size of each packet is considered to be 2000 bytes with a packet data rate 22.69 Mb/s. The generated data packet at each SU follows the Poisson process of arrival, where the multiple queues are maintained by each SU for its one-hop neighbors. In the proposed AACH model, the destination SU is decided when the RTS sent by the source SU is cleared by the CTS during the rendezvous. After the successful exchange of RTS/CTS between both SUs, the data transmission continues during the subsequent time slots, where the rendezvous occurs. The study of the impact of a number of channels on throughput, ATTR,

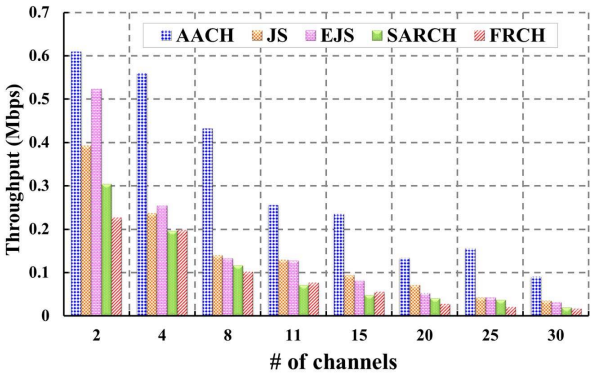


Fig. 8. Throughput versus number of channels.

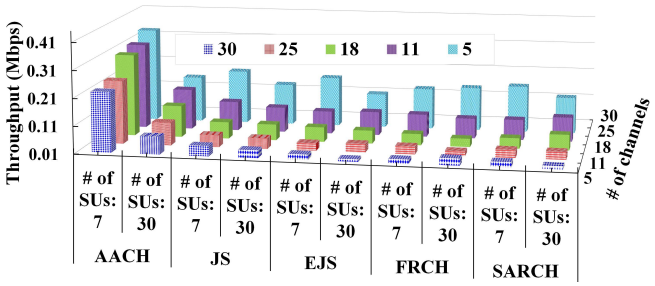


Fig. 9. Throughput versus number of channels versus number of SUs.

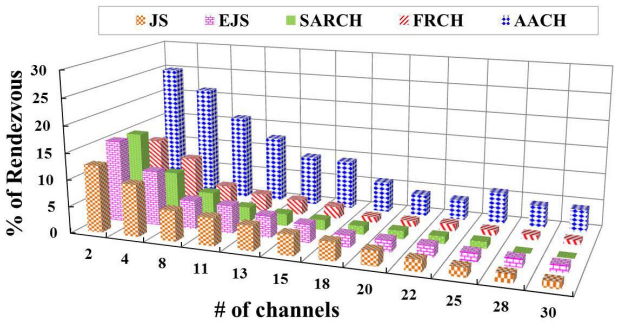


Fig. 10. Percentage of Rendezvous versus number of channels.

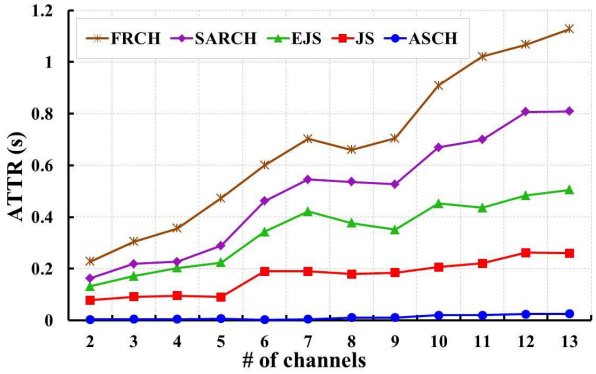


Fig. 11. ATTR versus number of channels.

and % of rendezvous in the asynchronous environment is performed for both two-user and multiuser scenarios considering asymmetric channel model.

The impact of throughput over number of channels is shown in Fig. 8, where AACH outperforms over others as it guarantees at least $2(|N|+1)$ number of rendezvous in the asymmetric asynchronous scenario. In case of both JS and EJS, there is no guaranteed rendezvous if both SUs choose different stay channel and random common available channel in the jump pattern. Although there is guaranteed rendezvous in FRCH and SARCH, the rendezvous percentage is very low resulting lower throughput. Furthermore, AACH shows better performance in terms of throughput even in asynchronous condition by introducing the SS mechanism.

In Fig. 9, the impact of throughput over number of channels in a multiuser scenario is evaluated. The throughput of JS and EJS gradually decreases in the multiuser scenario due to less chance of getting a common available channel in the jump pattern as both protocols adopt the random selection of the commonly available channels. Similarly, in case of SARCH and FRCH, although the introduction of adaptive CH_Seq mechanism creates the opportunity for rendezvous, the degree of rendezvous is very small as the replacement of common available channel varies with the selected rotation seed. However, in the proposed AACH protocol, the throughput in the multiuser scenario is very high due to collision-free rendezvous as two or more SUs can have a plenty of choice for selection of common sequences. Besides, AACH guarantees at least $2(|N|+1)$ numbers of rendezvous in a CH_Seq.

From Fig. 10, it is observed that the percentage of rendezvous in AACH is very high in comparison to other protocols. In AACH, there is at least $2(|N|+1)$ number rendezvous in a CH_Seq and maximum degree of rendezvous is $\{(|N|+1)^2 - ([|N|+2]/2)\}$. In contrast, the percentage of rendezvous in case of JS and EJS gradually decreases due to the random selection of the common available channels in the jump and stay pattern creating a lower chance of rendezvous in a common available channel during the CH. It is observed that percentage of rendezvous in SARCH and FRCH is $\approx (2|N|/[(2N+1)^2])$ and $(4/[(2N+1)])$, respectively, even after implementing the concept of adaptive sequences, which is very low.

The impact of ATTR over number of channels is evaluated as shown in Figs. 11 and 12. ATTR of AACH is very small, i.e., $<(|N|+1)$ with the availability of the single common channel, which can be improved with an increase in the number of available channels. However, in both JS and EJS, the value of ATTR is $> 3P$ and $> 4P$, respectively, where $P > N$. In case of SARCH and FRCH, the ATTR value increases with the number of channels as the generation of CH_Seq depends on the value of the rotation seed. Hence, it takes a longer time for both SUs to find a common available channel, if both of them choose different rotation seeds. Furthermore, the MTTR in SARCH and FRCH is very large, i.e., $> 4N$ and $> 2N+1$, respectively.

The impact of throughput and ATTR over different clock drift values with the fixed number of channels $|N| = 30$ are shown in Fig. 13(a) and (b), respectively. The clock drift value is considered as the entry slot index of the SUs to the network. From Fig. 13(a), it is observed that AACH performs better as

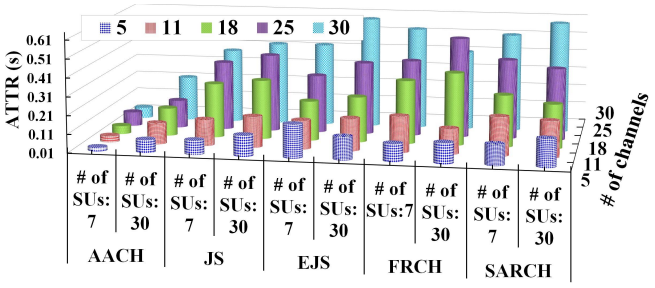
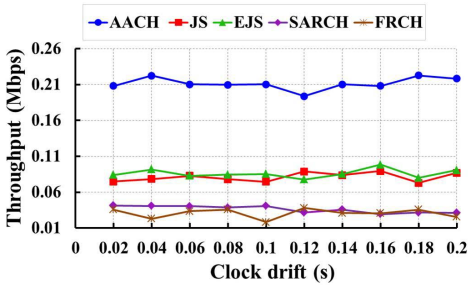
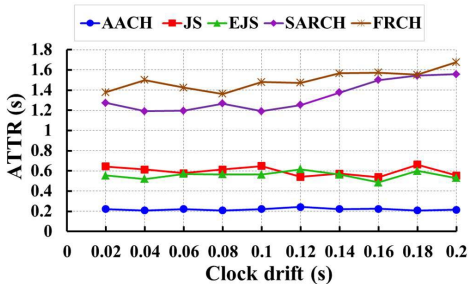


Fig. 12. ATTR versus number of channels versus number of SUs.



(a)

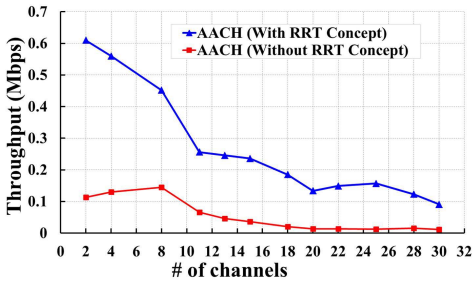


(b)

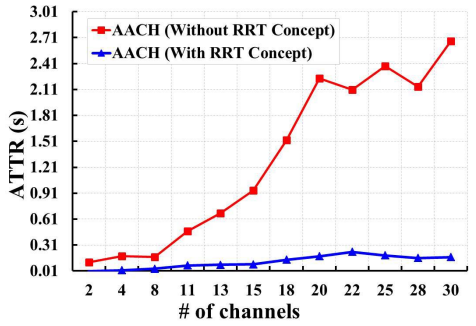
Fig. 13. Performance comparison between AACH with other protocols considering clock drift values. (a) Throughput of rendezvous versus clock drift. (b) ATTR versus clock drift.

compared to other protocols even with variable clock drift values. In AACH, the throughput gradually increases for the clock drift values > 0.12, whereas it gradually decreases or almost remains constant for other protocols. Based on different clock drift values, the jump and stay pattern of the SU varies, thereby resulting in less number of rendezvous in case of JS and EJS. In case of SARCH and FRCH, the common available channels are replaced with the unavailable channels in a CH_Seq. Hence, the SUs with different clock drift values have a higher chance to meet in their personal available channel instead of a common available channel. However, AACH outperforms over other protocols as there must be at least $2(|N|+1)$ number of rendezvous in any scenario.

As shown in Fig. 13(b), it is observed that the ATTR value of AACH is less as compared to other protocols. In case of JS, EJS, FRCH, and SARCH, the ATTR value is very large, i.e., $> |N|$ and almost constant for different clock drift values. Nevertheless, in AACH the value of ATTR is $<(|N|+1)$, which enables smaller waiting time for rendezvous between the SUs.



(a)



(b)

Fig. 14. Performance comparison between AACH with and without RRT concept. (a) Throughput versus number of channels. (b) ATTR versus number of channels.

In order to compare the performance of AACH with and without RRT concept, simulations are performed for the throughput and ATTR as shown in Fig. 14(a) and (b), respectively. It is to be noted that the CH_Seq consists of the common sequences in case of AACH with RRT, which is modified by using the concept of RRT. In the second scenario, CH_Seq is created by using only the permuted common sequence without the use of the RRT concept. In the latter scenario, the common available channel appears maximum two times in the common sequences except for s_0 and $s_{|N|+1}$. However, due to the slot misalignment in the asynchronous scenario, SUs get a very rare chance to meet in that two numbers of common available channels resulting in gradual decrease of the throughput. In contrast, the number of rendezvous in each common sequence is at least 4 after common channel replacement using RRT in the scenario of AACH with RRT concept. Since, there is no guaranteed time interval for rendezvous without RRT in AACH, larger MTTR value is observed as shown in Fig. 14(b). However, the AACH with RRT guarantees the rendezvous within $2(|N|+1)$ number of slot indexes after applying SS mechanism.

VII. Conclusion

In this paper, a new AACH algorithm is designed for IoT-based CRNs to achieve the guaranteed rendezvous within a bounded interval of time by using the concept of RRT. Furthermore, a novel concept of SS mechanism is introduced in the proposed protocol to minimize the MTTR. Theoretical analysis of AACH protocol is made to justify the correctness and to measure its performance. Performance evaluation of AACH indicates that it can outperform in terms of degree

of rendezvous, ATTR and throughput as compared to other protocols. Though, higher degree of rendezvous is bit challenging in asymmetric asynchronous environment, AACH can provide 100% guaranteed rendezvous in a CH_Seq. Hence, the proposed protocol can be highly useful for CR enabled IoT devices to utilize the unused spectrum efficiently. The proposed AACH protocol guarantees the rendezvous between two-users as well as multiusers considering the one-hop scenario. However, to overcome the limitations of data transmission range due to low bandwidth, low data rate, and signal fading, the proposed work can be extended to multihop communication scenario. Furthermore, SUs equipped with multiradios can be implemented.

REFERENCES

[1] S. Aslam, W. Ejaz, and M. Ibnkahla, "Energy and spectral efficient cognitive radio sensor networks for Internet of Things," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3220–3233, Aug. 2018.

[2] H. A. B. Salameh, S. Almajali, M. Ayyash, and H. Elgala, "Spectrum assignment in cognitive radio networks for Internet-of-Things delay-sensitive applications under jamming attacks," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1904–1913, Jun. 2018.

[3] G.-Y. Chang, W.-H. Teng, H.-Y. Chen, and J.-P. Sheu, "Novel channel-hopping schemes for cognitive radio networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 2, pp. 407–421, Feb. 2014.

[4] P. K. Sahoo, S. Mohapatra, and J.-P. Sheu, "Dynamic spectrum allocation algorithms for industrial cognitive radio networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3031–3043, Jul. 2018.

[5] A. A. Khan, M. H. Rehmani, and A. Rachedi, "When cognitive radio meets the Internet of Things?" in *Proc. IEEE Conf. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Sep. 2016, pp. 469–474.

[6] P. K. Sahoo and D. Sahoo, "Sequence-based channel hopping algorithms for dynamic spectrum sharing in cognitive radio networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 11, pp. 2814–2828, Nov. 2016.

[7] S. Mohapatra and P. K. Sahoo, "ASCH: A novel asymmetric synchronous channel hopping algorithm for cognitive radio networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2016, pp. 1–6.

[8] X. J. Tan, C. Zhou, and J. Chen, "Symmetric channel hopping for blind rendezvous in cognitive radio networks based on union of disjoint difference sets," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10233–10248, Nov. 2017.

[9] J.-P. Sheu and J.-J. Lin, "A multi-radio rendezvous algorithm based on Chinese remainder theorem in heterogeneous cognitive radio networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 9, pp. 1980–1990, Sep. 2018.

[10] C.-S. Chang, G.-C. Yang, M.-H. Chiang, and W. C. Kwong, "Construction of synchronous-symmetric channel-hopping sequences over Galois extension field for cognitive radio networks," *IEEE Commun. Lett.*, vol. 21, no. 6, pp. 1425–1428, Jun. 2017.

[11] J.-P. Sheu, C.-W. Su, and G.-Y. Chang, "Asynchronous quorum-based blind rendezvous schemes for cognitive radio networks," *IEEE Trans. Commun.*, vol. 64, no. 3, pp. 918–930, Mar. 2016.

[12] J. Li, H. Zhao, J. Wei, D. Ma, and L. Zhou, "Sender-jump receiver-wait: A simple blind rendezvous algorithm for distributed cognitive radio networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 1, pp. 183–196, Jan. 2018.

[13] X. Liu and J. L. Xie, "Priority-based spectrum access in cognitive D2D networks for IoT," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[14] G.-Y. Chang and J.-F. Huang, "A fast rendezvous channel-hopping algorithm for cognitive radio networks," *IEEE Commun. Lett.*, vol. 17, no. 7, pp. 1475–1478, Jul. 2013.

[15] I.-H. Chuang, H.-Y. Wu, and Y.-H. Kuo, "A fast blind rendezvous method by alternate hop-and-wait channel hopping in cognitive radio networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 10, pp. 2171–2184, Oct. 2014.

[16] H. Liu, Z. Lin, X. Chu, and Y.-W. Leung, "Jump-stay rendezvous algorithm for cognitive radio networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 10, pp. 1867–1881, Oct. 2012.

[17] Z. Lin, H. Liu, X. Chu, and Y.-W. Leung, "Enhanced jump-stay rendezvous algorithm for cognitive radio networks," *IEEE Commun. Lett.*, vol. 17, no. 9, pp. 1742–1745, Sep. 2013.

[18] W.-C. Chen, G.-C. Yang, M.-K. Chang, and W. C. Kwong, "Construction and analysis of shift-invariant, asynchronous-symmetric channel-hopping sequences for cognitive radio networks," *IEEE Trans. Commun.*, vol. 65, no. 4, pp. 1494–1506, Apr. 2017.

[19] G.-Y. Chang, J.-F. Huang, and Y.-S. Wang, "Matrix-based channel hopping algorithms for cognitive radio networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 5, pp. 2755–2768, May 2015.

[20] M. Monemi, M. Rasti, and E. Hossain, "On characterization of feasible interference regions in cognitive radio networks," *IEEE Trans. Commun.*, vol. 64, no. 2, pp. 511–524, Feb. 2016.

[21] T. Koshy, *Elementary Number Theory With Applications*. Amsterdam, The Netherlands: Academic, May 2007.