

An Asynchronous Neighbor Discovery Algorithm for Cognitive Radio Networks

Short Paper

Chanaka J. Liyana Arachchige, S. Venkatesan and Neeraj Mittal
Erik Jonsson School of Engineering and Computer Science
{chanaka.liyana, venky, neerajm}@utdallas.edu
University of Texas at Dallas, Richardson, TX 75080

Abstract - Cognitive Radio (CR) technology is a promising emerging technology that enables a more efficient usage of already assigned spectrum. However, before CR networks can be deployed, new algorithms and protocols need to be developed. When forming an ad-hoc Cognitive Radio Network (CRN), one of the fundamental tasks is to determine the neighbors of each node and channels that can be used to communicate among neighbors. The nature of the CRN makes this a challenging problem. Specifically in CRN, not only the network is multi-channel, but the channels available at different nodes may be different. Furthermore no time synchronization may be available to nodes. In this paper, we propose an asynchronous distributed algorithm that allows nodes to discover their neighbors and channels that can be used to communicate with them in a single-hop CRN.

I. INTRODUCTION

Cognitive Radio (CR) enables the use of unused spectrum in opportunistic manner. It allows nodes to sense the spectrum over wide range of bands and use a channel only if it does not cause interference to the licensed user. Licensed user is the owner of the channel and is referred to as the *Primary User*. All other users of the channel are referred to as *Secondary Users* [1]. When the primary user returns to the channel, CR nodes need to vacate the channel and switch to some other available channel for communication.

In a Cognitive Radio Network (CRN), node A and node B are said to be neighbors if both A and B are within transmission range of each other and have at least one common channel between them. Each node is capable of periodically scanning and identifying the *available channel set* for the node. Channel C is said to be available if a CR node can transmit and receive on C for reasonable amount of time without causing/having interference to/from primary users [3].

A. Motivation

In order to form and configure a CRN, nodes need to discover their neighbors and usable channels to communicate among them. But individual nodes may not have any prior knowledge about their neighboring nodes and the network topology. This is certainly the case in situations like military and first responders networks, where early application of CR technology is expected to appear [2]. Due to the geographical separation between nodes, different CR nodes may have different *available channel sets*. Over time, this *available channel set* can change due to the arrival and departures of primary users. Also, new CR nodes can join the network at any

time. Further time synchronization may not be available to all the nodes.

These properties of CRN require new algorithms to form and configure CR networks. Any algorithm that addresses the above problem requires answering the following questions.

- a. Who are the neighbors of each node?
- b. What are the channels that can be used to communicate among neighbors?

Our objective is to develop an effective algorithm that provides answers to the above questions.

B. Related work

Even though neighbor discovery has been well studied for single channel ad-hoc networks, not much work has been done for multi channel networks.

Srinivasan et al. [4], [3] have proposed several neighbor discovery and configuration algorithms for CR networks considering single and multi transceiver CR nodes. But these algorithms need nodes to be synchronized with each other and to start certain tasks at the same time.

Assuming a presence of a control channel, Thoppian et al. [5] proposed an algorithm for neighbor discovery, MAC layer configuration, and routing for CR networks. This work also assumes the availability of time synchronization between nodes.

Asynchronous, multi-channel neighbor discovery algorithms have been developed by the Bluetooth research community [6],[7]. But they address a case where channel set is fixed, while in CRN channel set can vary among nodes. Also in Bluetooth, neighbor discovery is asymmetric (master/slave configuration), but in CRN we consider a symmetric scenario. In fact, we have borrowed some ideas from Bluetooth neighbor discovery process when developing our algorithm.

C. Our Contribution

In this paper we propose an *asynchronous* neighbor discovery and configuration algorithm for a single hop CRN. Even though neighbor discovery and configuration problem have been addressed for time synchronized CR networks, we believe this to be the first work that addresses the neighbor discovery and configuration problem for *asynchronous* CR networks. In summary, our main contributions are:

- We propose a distributed algorithm to elect a leader for the CRN.

- The algorithm enables the leader node to discover the neighboring nodes and their channel sets.
- The algorithm allows the leader node to find the globally common channel set.
- We provide a way to allow the leader node to inform other nodes about their neighbors, channel sets, and globally common channels.

The rest of the paper is organized as follows. Section 2 presents the system model and assumptions. Section 3 provides the details of the algorithm. In section 4 we discuss the correctness of the algorithm. Section 5 presents an analysis of the algorithm and section 6 concludes the paper.

II. SYSTEM MODEL

We assume a “fully connected” wireless network, consisting of CR nodes. By fully connected, we mean that all the nodes reside within the coverage area of each other and each node has at least one common channel with its neighbors.

We do not require time synchronization between CR nodes. But we assume clock ticking speeds of different nodes to be comparable. It means that when node B receives a clock value from node A , node B can synchronize with node A , using A 's clock value. There is no globally common clock for the network.

CRN has M channels available for its operations. Let this channel set be $C_{global} = \{C_1, C_2, \dots, C_M\}$.

Each node in CRN is assigned a unique identifier (Node ID) in the range $1 \dots N$. Each node knows its unique Node ID, but does not know how many other nodes there are and what their IDs are. Nodes are equipped with one transceiver (Transmitter & Receiver) capable of either receiving or transmitting at any given time. Nodes know about the global channel set C_{global} and the value of N . Also each node finds its set of usable channels (*available channel set*) by a periodic scanning mechanism. We assume nodes can detect collisions.

III. NEIGHBOR DISCOVERY ALGORITHM

In this section, first we will give a simple overview of the algorithm. Then we will describe the modes and time durations used in the algorithm. Finally we will give a detailed description of the algorithm.

A. Overview of the Algorithm

The algorithm is based on electing a leader and then the leader node performing neighbor discovery and informing other nodes about their neighbors.

When a new node (say Node A) is ready to join the network, it first checks if a leader already exists or whether some nodes are already trying to become the leader. This is done by listening on all available channels for sufficiently large amount of time. If a leader is detected, node A let leader node to discover itself. If node A hears from a node which is trying to become the leader, node A back off and waits for the leader to be elected.

If node A does not hear from any existing nodes, it tries to become the leader. If the attempt is successful, it elects itself as the leader. Otherwise it waits until a leader is elected.

Once a node is elected as the leader, it executes neighbor discovery procedures. Then the leader starts normal network operations. During the normal operations, leader periodically performs neighbor discovery in order to discover newly arrived nodes. After each neighbor discovery period, the leader node informs all discovered nodes about their neighbors and neighbor's *available channel sets*.

B. Modes

Before going into details we need to define few modes used in the algorithm.

1) Scan Mode(SM)

In SM, a node listens on each of the channel in its *available channel set*, one-by-one. Receiving duration on each channel is $2T_b(M+1)$. (T_b is defined later in Contending Mode section.) Figure 1 shows SM operation for a node with available channel set $\{1, 2, 3\}$.

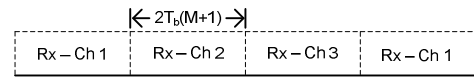


Figure 1: Scan Mode example

2) Contending Mode(CM)

CM is used by nodes to announce their intention to become the leader. A node in this mode continuously transmits beacons called *Contending Beacon (CB)* on each channel from its *available channel set* successively. Beacon duration is T_b and there is a T_b idle period between two successive beacons. Each beacon contains the ID of the sending node. Figure 2 shows an example of CM for a node where availability set is $\{1, 2, 3\}$.

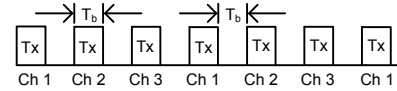


Figure 2: Contending Mode example

3) Inquiry Mode (IM)

This mode is used by the leader node to discover neighboring nodes. A node in this mode transmits and receives alternatively on each channel of its *available channel set*. During the transmit period it sends a beacon referred to as *Inquiry Beacon (IB)* with node ID, clock value of the node, available channel set and remaining time in IM. The beacon duration is T_b .

During the receiving period, inquiring node listens on the same channel that it transmitted previously, for duration of T_b . If a valid reply (*Inquiry Reply*) is received, the inquiring node sends an *Acknowledgment (Ack)* message, during the current receiving period or during the next receiving period, on the same channel that it received the reply. Upon receiving an *Inquiry Reply* message, inquiring node discovers the sender of the *Inquiry Reply*. Figure 3 shows an IM operation for a node with available channel set $\{1, 2, 3\}$.

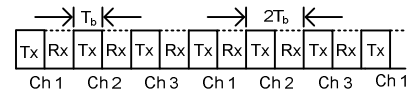


Figure 3: Inquiry Mode example

4) Inquiry Reply Mode (IRM)

This mode is used by the nodes in the SM to send reply messages to the inquiring node. Upon receiving an *IB* message from inquiring node, scanning node goes to IRM and sends an *Inquiry Reply (IR)* message to the inquiring node. Using the information contained in the *IB* and depending on the replying node's *available channel set*, replying node calculates possible time slots to send the reply. Then the replying node randomly selects a slot (in order to avoid collisions) and sends an *IR* during that slot. Each *IR* message contains the ID and *available channel set* of the sender.

After sending an *IR* message, node waits on the same channel for $3T_b$ time period in order to receive an *Acknowledgement (Ack)* from the inquiring node. This $3T_b$ period is due to the fact that inquiring node can send *Ack* during the current receiving period or next receiving period which comes after a transmitting period. If an *Ack* is successfully received, node in the IRM knows that it was discovered successfully. It then goes to the Wait Mode. If no *Ack* is received, node assumes the discovery was unsuccessful and sends another *IR* during the next possible slot. After trying a certain number of times, node switches back to SM.

5) Wait Mode (WM)

Once a node is successfully discovered, it goes to WM. During the WM, the node continuously listens on the same channel that it was discovered (the channel on which *IR* is sent).

C. Time Duration Selection

1) Scan Duration

In the SM, the receiving period of each channel is $2T_b(M+1)$. This selection allows a node in either IM or CM to transmit beacons on all the available channels during a single scan period. (Time between two consecutive beacons is $2T_b$. So $2T_b(M+1)$ allows receiving of M complete beacons.) Figure 4 show an example scenario assuming a global channel set of 3 ($M = 3$).

2) T_l Time Duration

T_l is an important parameter used by the algorithm. Most of the operations in the algorithm used this as a base time unit. It is define as follows:

Assume node *A* is in IM and node *B* is in SM. T_l is the overlap time needed between *A* and *B* in order to guarantee that node *B* will detect an *IB* message from node *A*. (Assuming the presence of a common channel.) T_l is given by following equation.

$$T_l = 2T_b(M+1)M$$

For example assume a network with 3 channels ($M = 3$). Also consider two nodes (*A* and *B*) in the network with *available channel sets* $A = \{3\}$, $B = \{1,2,3\}$. *A* is in IM and *B* is in SM. The T_l duration is shown in the figure 4.

According to the definition of T_l , when one node is in SM for T_l time and other node is in IM for same T_l duration or more, scanning node is guaranteed to discover inquiring node provided no collisions occur. This is also true if one node is in SM and other is in CM.

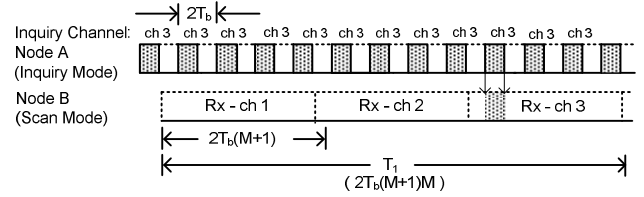


Figure 4: T_l time duration

D. Algorithm Description

Operations of the algorithm can be categorized into four main phases. Figure 5 shows detailed view of these phases with respect to the leader node.

1. Leader Detection Phase (LDP)
2. Leader Election Phase (LEP)
3. Neighbor Discovery Phase (NDP)
4. Normal Operations Phase (NOP)

1) Leader Detection Phase (LDP)

When a CR node is ready to join the network, it tries to detect any existing leader node or nodes who are trying to become the leader by entering the LDP. During this phase, node enters the SM and scans for sufficiently larger amount of time in order to detect any beacon transmitting nodes.

If there is already elected leader in the network, the scanning node should hear an *IB* message from the leader. Then the scanning node goes to IRM and replies to the *IB* message. If there are nodes contending for the leadership, the scanning node should receive *CB* messages. In that case the scanning node continues to stay in the SM until a leader is elected and an *IB* message is received.

There is a possibility of *collisions* due to *CB* and *IB* messages. If a scanning node detects a *collision*, it can safely assume the presence of other CR nodes in the network. Then scanning node should continue to stay in the SM until a leader is elected and an *IB* message is received. (Collisions occurring from primary nodes are not considered here.)

If nothing is received during the scanning period, the scanning node enters the Leader Election Phase.

Duration of LDP is aT_l where a is a positive integer and T_l is the base time defined previously. We select this time in order to guarantee detection of the leader when present. When a leader is elected, it performs periodic inquiry operation for neighbor discovery. This is done for duration of T_l for each T time units. Therefore if the scanning node scans for T duration, it is guaranteed to have overlap duration of T_l between the leader and scanning node. This guarantees the detection of the leader node (or collisions) by the scanning node. Therefore the duration of the LDP (aT_l) should be equal to T .

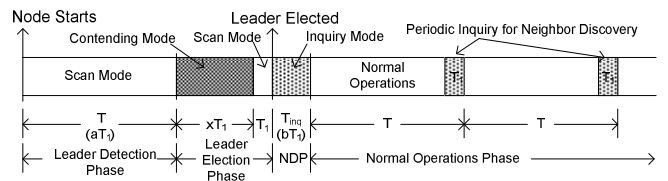


Figure 5: Stages of the algorithm with respect to the leader.

2) Leader Election Phase (LEP)

LEP consists of two mode of operations; Contending Mode and Scan Mode.

A node in the LEP first enters the CM and transmits *CB* messages to announce its intention to become the leader. There are two goals associated with this mode.

1. Prevent other nodes from entering the LEP.
2. Provide a way to break the tie among competing nodes.

When a scanning node receives a *CB*, it does not try to become the leader node. Therefore the first task is achieved by using *CB* messages. In order to achieve the second task, the algorithm associates node IDs with the CM. This is done by using a unique value (say x) derived from node ID to determine the CM duration. Then CM duration is given by xT_l . As an example if we have a node with ID 10, we can select CM duration to be $10T_l$.

After staying in the CM for xT_l duration, a node again goes to SM. If nothing is received for a period of T_l , the node elects itself as the leader. If the node receives a *CB* message or detects a *collision* during this period, it continues to stay in the SM, until it receives an *IB* from a future leader. If it receives an *IB* message, the node changes to IRM.

According to our leader election process, a node with higher ID will be elected as the leader, if they started at the same time and no other nodes are present in the network. If they started at different times, the leader would be determined based on the starting times and node ID's. Figure 6 shows an example for leader election with 2 nodes; node A and node B.

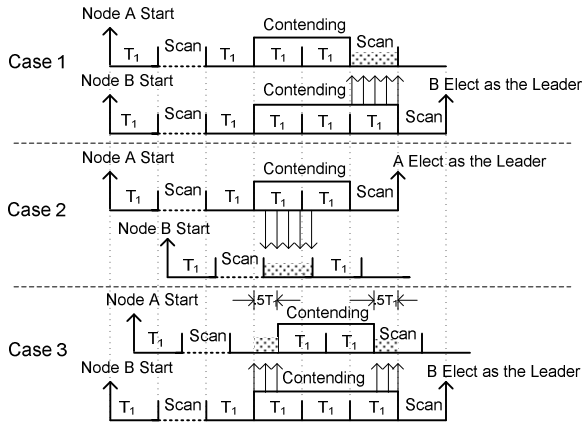


Figure 6: Leader election example with two nodes

We have presented 3 cases in the above example. Node A's ID is 2. Node B's ID is 3. Their CM durations are $2T_l$ and $3T_l$ respectively. In case 1, both nodes begin at the same time. Since B's ID is higher, it is elected as the leader.

In case 2, node B begins after A. In this case, there is a complete T_l overlap time between A's transmission and B's reception. Therefore B should receive at least one *CB* message from A. So node B backs off and node A is elected as the leader.

In case 3, node A starts $0.5T_l$ later than node B. As a result, there will be a $0.5T_l$ overlap time between the node A's SM and node B's CM. If node A successfully receives a *CB* message during this period, it will back off and let node B

become the leader. If node A does not receive a *CB* during this period, node A enters into CM and then to SM. During this second SM, again there will be a $0.5T_l$ overlap time between node A's SM and node B's CM. Since there is T_l total overlap time between two nodes, node A should receive node B's *CB*. Therefore node A will back off and let node B elects itself as the leader.

3) Neighbor Discovery Phase (NDP)

Once a node is elected as the leader, it starts the NDP. During this phase, the leader stays in IM and transmits and receives alternatively. During the transmission periods, it sends *IB* messages, allowing scanning nodes to discover the leader. Upon receiving *IB* messages, the scanning nodes change to IRM and send replies (*IR*) to the leader. When the leader receives an *IR* from a node, the leader discovers the sender of the *IR* message. Then the leader sends an *Ack* message to the discovered node, confirming the discovery. Once a node receives an *Ack* message, the node considers itself as discovered and goes to WM.

The leader stays in the NDP for bT_l duration where 'b' is a positive integer. Clearly, all scanning nodes should be able to discover the leader node, during a period of T_l , provided no *collisions* occur. To reduce the possibility of replies collided with each other, scanning nodes do not reply to *IB* immediately. They calculate the possible time slots (receiving periods of the leader node) to send the reply (*IR*) and randomly select a time slot from the results. In order to give a wider time window for scanning nodes to select these timeslots, we suggest selecting 'b' to be 2 or grater.

4) Normal Operations Phase (NOP)

After the NDP, the leader goes to NOP. All the normal communications in the network happen during this phase.

During normal operations, the leader periodically invokes the neighbor discovery process by changing to IM. This is done for a period of T_l , every T time units. This periodic neighbor discovery is done to discover nodes that were not discovered during the previous NDP and to discover newly arrived nodes.

After each neighbor discovery period (T_l), the leader informs already discovered nodes about their neighbors and their *available channel sets*.

During the periodic neighbor discovery, all discovered nodes should stay idle. Then they should go to WM in order to receive messages from the leader about their neighbors and neighbor's *available channel sets*. Then they can join the NOP.

5) Selection of T

The value of T needs to be selected in order to maximize the available time for normal operations. For example if we select T to be $10T_l$, 90% of time will be available to normal operations. On the other hand if we select T to be very large, nodes may have to wait for a long time in LDP. (Recall LDP time is also equal to T).

1. When a node is turned on, go to SM.
2. If a *CB* or *collision* received, wait on SM for *IB* msg.
3. If an *IB* message received, change to IRM.
4. If no messages or *collisions* received for aT_1 time, change to CM. Stay in for xT_1 time. (x is a unique number generated from node ID). Then change to SM.
 - a. If a *CB* or *collision* received, wait on SM for *IB*.
 - b. If an *IB* received, change to IRM.
 - c. If no messages or *collisions* received during the T_1 period, elect itself as the leader. Then change to IM for T_{inq} time. Then to NOP.
5. In IRM, nodes calculate the possible time slots to send the reply. Then send an *IR* message on randomly selected slot. Then wait for an *Ack* for $3T_b$ time period.
 - a. If an *Ack* is received, stay on the same channel where the *IR* message was sent.
 - b. If no *Ack* is received, try sending *IB* for n times. If can not send, go to SM.

Figure 7: Neighbor Discovery Algorithm

IV. CORRECTNESS OF THE ALGORITHM

We have shown the correctness of the algorithm using the following two lemmas. Please refer to [8] for the proofs.

Lemma 1. *At least one node will be elected as the leader, within some finite amount of time. Also the starting time of this leader node happens within the time period $[0, T_1]$.*

Lemma 2. *No more than one node will be elected as the leader.*

V. ANALYSIS

In this section we derive an upper bound for time required for the leader to be elected.

Consider a CRN with N nodes and M channels. Each node has a unique ID, from the range 1 to N . Largest node ID is N . Let us assume that the very first node joins the CR network enters at $t = 0$.

Lemma 3. *Any node entering into network after $t = T_1$ will not be able to become the leader.*

Proof: From Lemma 1 at least one leader node will be elected and, moreover, the leader node starts during the period $[0, T_1]$. From Lemma 2, two nodes cannot be elected as leaders. Therefore any node joins after $t = T_1$ will not be able to become the leader. ■

From Lemma 3, any node that starts after $t = T_1$ will not be able to become the leader. Therefore the node which becomes the leader, should start on or after $t = 0$ and before $t = T_1$. Then by adding this initial T_1 duration with initial SM duration (aT_1), CM duration (Maximum duration: NT_1) and second SM duration (T_1), we get $(2 + a + N)T_1$ as the upper bound for the leader election process. Since $T_1 = 2T_b(M+1)M$, upper bound for leader election process is $O(NM^2)$. Also the time required for the normal operations phase to begin is $(2 + a + b + N)T_1$.

Example: Assume a scenario where CR technology is used to communicate among a 25 member first responders team. Also assume $|C_{global}| = M = 30$. Let $2T_b = 1$ millisecond, $a = 10$ and $b = 2$. Then we get T_1 to be 930ms. Worst case time required for normal operations to begin is 36.27s.

VI. CONCLUSION

In this paper we have proposed an asynchronous distributed algorithm for solving the neighbor discovery problem in fully connected CR networks. The algorithm is based on electing a leader for the network and then the leader performing neighbor discovery operations. Also the algorithm provides a way of informing discovered nodes about their neighbors and neighbor's available channel sets. We have also derived an upper bound for the time required for the leader to be elected and to discover neighbors. This upper bound is given by $O(NM^2)$ timeslots where N is the maximum number of nodes and M is the maximum number of channels available for the network. Since our algorithm elects a leader and then the leader collects all the information regarding the neighbors and their available channels, other necessary network operations such as scheduling, network management, etc. can be done easily using the elected leader.

REFERENCES

- [1] R.W. Brodersen et al.. "Corvus: A cognitive radio approach for usage of virtual unlicensed spectrum".
Webpage - <http://citeseer.ist.psu.edu/756492.html>
- [2] Shared Spectrum Company – Press release - "Shared Spectrum Company Awarded Funding to Develop Cognitive Radio System for First Responders".
<http://www.sharedspectrum.com/news/announcements/pr101507.html>
- [3] S. Krishnamurthy, R. Chandrasekaran, N. Mittal, and S. Venkatesan, "Synchronous Distributed Algorithms for Node Discovery and Configuration in Multi-channel Cognitive Radio Networks", in DISC 2006, September 2006
- [4] S. Krishnamurthy, M. Thoppian, S. Kuppa, R. Chandrasekaran, N. Mittal, S. Venkatesan and R. Prakash, "Time-efficient Distributed Layer-2 Auto-configuration for Cognitive Radio Networks," Computer Networks, Vol 52, Issue 4, 2008.
- [5] M. Thoppian et al. "Control Channel-based MAC-layer Configuration, Routing and Situation Awareness for Cognitive Radio Networks", in MILCOM 2005, October 2005.
- [6] B.S. Peterson, R.O. Baldwin, J.P. Kharoufeh, "Bluetooth Inquiry Time Characterization and Selection" on IEEE Transaction on Mobile Computing, Vol. 5, No. 9, Sept. 2006.
- [7] T. Salonidis, P. Bhagwat, L. Tassiulas, R. LaMaire, "Distributed Topology Construction of Bluetooth Personal Area Networks" In IEEE INFOCOM, April 2001.
- [8] C. Liyana Arachchige, S. Venkatesan and Neeraj Mittal, "Asynchronous Neighbor Discovery Algorithm for Cognitive Radio Networks", Technical Report UTDCS-24-08, University of Texas at Dallas, August 2008.