Virtual Backbone Construction for Cognitive Radio Networks without Common Control Channel

Ying Dai,† Jie Wu,† and ChunSheng Xin‡
Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122†
Department of Computer Science, Norfolk State University, Norfolk, VA 23504‡

***Abstract*—The advantages of virtual backbones have been proven in wireless networks. In cognitive radio networks (CRNs), virtual backbones can also play a critical role in both routing and data transport. However, the virtual backbone construction for CRNs is more challenging than for traditional wireless networks because of the opportunistic spectrum access. Moreover, when no common control channel is available to exchange the control information, this problem is even more difficult. In this paper, we propose a novel approach for constructing virtual backbones in CRNs, without relying on a common control channel. Our approach first utilizes the geographical information to let the nodes of a CRN self-organize into cells. Next, the nodes in each cell form into clusters, and a virtual backbone is established over the cluster heads. The virtual backbone is then applied to carry out the end-to-end data transmission. The proposed virtual backbone construction approach requires only limited exchange of control messages. It is efficient and highly adaptable under the opportunistic spectrum access. We evaluate our approach through extensive simulations.**

***Index Terms*—Cognitive radio networks, self-organization, virtual backbone construction, end-to-end data transmission.**

I. Introduction

In *cognitive radio networks* (CRNs) [1], nodes can make opportunistic use of multiple channels that are not occupied by primary users. However, when a primary user begins to occupy a channel, secondary users on that channel need to quit immediately. Hence, the dynamics of channel availability makes it difficult to carry out end-to-end data transport in CRNs. If a node in a CRN wants to reach another node that is multiple hops away, two problems arise. First, the node needs to calculate the route to the destination node, which consists of a list of intermediate nodes. However, the high dynamics of channel availability makes it costly to collect information from other nodes and construct a routing path. Second, even if the route is built, the links on the route are unstable. When the dynamic channels on a link of the route become unavailable, the route is broken.

To solve the problem, we can make use of the virtual backbone structure [2]. A virtual backbone consists of a connected subset of nodes in the network where every node is either in the subset or a neighbor of a node in the subset. We use *area* to refer to a backbone node and the nodes attached to it. If a virtual backbone is constructed for a CRN, the backbone nodes can calculate area routes for end-to-end communications. An area route means a set of areas that would be passed in order to reach the destination. For example, in Fig. 1, each node is either a backbone node or attached to a backbone node. A_1 denotes an area, which includes the backbone node and its attached nodes. Nodes on the borders are called *gateway* nodes. The source node S wants to reach the destination node D, which is located in another area. The backbone node that S is attached to calculates an area route for S, which is $A_1 \rightarrow A_2 \rightarrow ... \rightarrow A_k$. Moreover, the virtual backbone can solve the unstable link problem, because with the area route, a packet can be sent to any node in the next-hop area. This is much more robust than the case with the route consisting of nodes, where a packet must be sent to the next-hop node. Therefore, the influence of unpredictable channel availability is reduced.

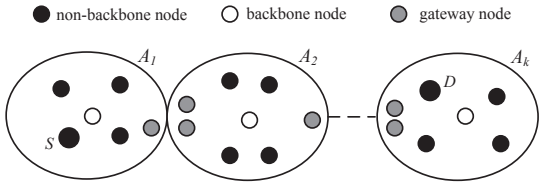


Fig. 1. Example of the end-to-end transmission using virtual backbone.

However, the virtual backbone construction in traditional wireless networks relies on a control channel to exchange extensive control information during the virtual backbone construction. The dynamic availability of channels in CRNs make it impractical to use a static channel to exchange control information between nodes. While many studies on CRNs assume a *common control channel* (CCC), it is vulnerable to jamming attacks and congestion. Furthermore, in order for the spectrum authorities to allocate a static band as the control channel for CRNs, it is often involved with international negotiations, which are very time consuming. Therefore, we need to find an efficient way for nodes in CRNs to form a virtual backbone without a CCC.

In this paper, we propose a novel approach for virtual backbone construction in a CRN without relying on a CCC. We use nodes to refer to secondary users. The cognitive radios of nodes are assumed to have the GPS waveform, so that each node knows its geographical location. We make use of the location information of each node and select channels for distributed control message exchange. Here, we apply a cell division methodology and assign active/passive states to

The research of Jie Wu is supported in part by NSF under grants ECCS 1231461, ECCS 1128209, CNS 1138963, CNS 1065444, and CCF 1028167. The research of ChunSheng Xin is supported in part by NSF under grants CNS 1217668, CNS 1017172, and ECCS 1247853.

each node. Through an efficient process of message exchange, cluster heads are selected based on cells. Then, we efficiently construct a virtual backbone by selecting backbone nodes from the cluster heads. After the virtual backbone is constructed, it is used for the end-to-end data transmission in a CRN. The virtual backbone node calculates the area route for a source node. Our approach supports simultaneous transmission of multiple communicating node pairs in an area, which is different from the data transmission in a virtual backbone of traditional wireless networks, in which individual nodes communicate only with the backbone node. Reliability and throughput are improved in our approach in comparison with CRNs without a backbone, as well as, traditional wireless networks with backbone. To avoid cochannel interference among different links, we propose an algorithm that chooses a different transmission channel for each node pair communicating simultaneously.

The main contributions in our work are as follows.

- To our best knowledge, this is the first work to apply the technique of virtual backbone to the CRN, to reliably and efficiently transport data in CRNs.
- We provide an algorithm for CRN self-organization with distributed control message exchange without CCC.
- We develop an approach to construct a virtual backbone for a CRN with limited message exchange.
- We also propose a novel end-to-end data transmission scheme for nodes in CRNs using the virtual backbone.

The organization of our paper is as follows. In Section II, we discuss the related works. The system model is introduced in Section III. Section IV describes the self-organization algorithm for the CRN. Sections V and VI present the virtual backbone construction, and the end-to-end data transmission scheme. The performance evaluation is presented in Section VII. We conclude our paper in Section VIII.

II. Related Works

There are many studies on the virtual backbone construction in traditional ad hoc networks [3]–[6]. The authors in [3], [4] adopted clustering formation approaches to form a maximal independent set (MIS). In an ad hoc network, the set of cluster heads is an MIS, acting as a virtual backbone. There are disadvantages in these approaches, such as relative slow convergence, high redundancy or overhead, which make them hard to be applied in CRNs. Nodes in CRNs need to use available channels more efficiently. Constructing virtual backbones in CRNs requires a faster convergence and lower overhead. In [5], the authors used a clustering formation approach followed by a pruning and marking process, which is able to control the density. In addition, they applied an adjustable transmission range, which reduces the energy cost and the MAC layer contention. The special method of dealing with mobility in virtual backbone constrution is discussed in [6].

[7]–[9] proposed several data transmission protocols in CRNs with CCCs. Each node uses the CCC to negotiate an operational channel from the dynamically detected idle licensed channels. However, in practice, it is often impractical to find a CCC. Moreover, the cochannel interference in the CCC is also a challenging issue. In [10]–[14], several approaches to building links in CRNs without CCC were proposed. [10] proposed an algorithm with the nodes rendezvous time as a function of the number of channels, while the algorithm in [11] has rendezvous time as a function of the number of nodes. Both of them are based on the assumption of some pre-known network information. Authors in [12] proposed a rendezvous algorithm based on the quorum systems. It makes use of the overlap of any two generated channel sequences in quorum systems for nodes rendezvous. However, in this approach, many nodes need to compete for one rendezvous channel. A jump-and-stay model was proposed in [13]. It is a blind channel rendezvous model. The approach in [14] utilizes the channel diversity and allows all channels to be a control channel. Our proposed approach is different from the existing studies for several reasons. First, we do not require any pre-known node or channel information. Second, our approach can be applied to multihop networks to perform efficient routing and data transport, while most of existing studies focused on single-hop networks.

III. System Model

We consider a CRN with a set of nodes N. Each node is equipped with a GPS device and, therefore, is able to know its current location. We assume that the transmission range of each node is controllable. This can be achieved through adjusting the transmitting power. Let M denote the set of all available channels to the CRN. The set of available channels at each node is expected to be different from node to node, due to spectrum sensing imperfection and spatial diversity of channel availability. Therefore, not all channels in M are available at every node. We use M_i to denote the set of available channels at node i. We assume that the nodes on the same channel use an existing multi-access MAC protocol (e.g., the IEEE 802.11) to access this channel.

We treat the geographical location of the network as a rectangle and divide the network into a set of square cells, $C = \{c_k | k = 1, 2, 3, ...\}$. The length of each cell side is L. Each cell has a unique ID by its geographical location. Each node knows the information of the network field (the rectangle), and L. Since each node knows its location using GPS, it is able to calculate which cell it is located in. The length of L is related to the transmission range of each node: $L = \frac{\sqrt{2}}{4}R$, where R is the data transmission range of each node. We will explain this setting later. We will adjust the transmission range for virtual backbone construction. But when performing the end-to-end data transmission, the transmission range used by each node is R.

The objective of our model is to construct a virtual backbone without a CCC. Then, using the virtual backbone, the end-to-end data transmission can be efficiently conducted. Our approach contains three phases: 1) self-organization: nodes are spontaneously organized into cells and learn the information of other nodes in the same cell with limited control message exchange; 2) virtual backbone construction: cluster heads are

Algorithm 1 $Pro1(i, M(c_k))$, to compute IHC for node i

1. Set i as the seed for the pseudo-random number generator Z.
2. Let $\mathcal{Q} = M(c_k)$ // The channel segment for c_k
3. **repeat**
4. $k = Z(|\mathcal{Q}|)$ // Generate k such that $1 \leq k \leq |\mathcal{Q}|$
5. $q = \mathcal{Q}(k)$ // $\mathcal{Q}(k)$ is kth channel in $\mathcal{Q}$
6. $\mathcal{Q} = \mathcal{Q} \backslash \{q\}$ // Remove q from $\mathcal{Q}$
7. **until** $q \in M_i$
8. Return q // Selected IHC

selected from each cell and a subset of these cluster heads forms into a virtual backbone, which ensures both coverage and connectivity; 3) end-to-end data transmission: with the help of the virtual backbone, an efficient scheme for end-to-end data transmission is developed. We will introduce the above three phases in the following three sections.

IV. Self-Organization

A. IHC Selection

For the communication between nodes that have no information about each other initially, we define two states for each node: *active* and *passive*.

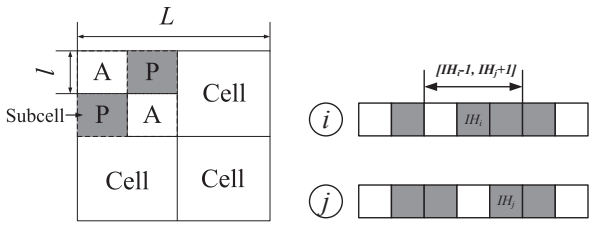
***Definition** 1:* A passive node is a node that keeps listening and receiving at a given channel. An active node is a node that guesses the channel that a passive node is on, and switches to that channel to send data packets. A node alternates between the active and passive states periodically.

For effective self-organization, it is critical to choose a channel on which a passive node can listen. We call such a channel for each node as an *initial home channel* (IHC), which is used for exchanging control messages initially. We describe how to select IHC next.

1) IHC Selection Algorithm: First, the whole channel set M is divided into $|C|$ segments. Cell c_k is assigned with a channel segment. We let $M(c_k)$ denote this segment of channels for cell c_k. Since each node is equipped with GPS, it is able to know its location and the cell that it is currently in. Thus, the node would also be able to know the channels segment that is assigned to its current cell. Nodes in the same cell would choose their IHCs from the segment of channels for the cell. For node i, its IHC is denoted as IH_i. The procedure $Pro1(i, M(c_k))$ for node i in cell c_k to obtain its IH_i is shown in Algorithm 1, which uses a similar approach as in [11].

2) Segment Size: We divide the whole available channel set M into $|C|$ segments. However, sometimes it is impractical to simply use $|M|/|C|$ to decide the number of channels for each segment. For example, if $|M| = 16$ and $|C| = 16$, then each cell is assigned with only 1 channel, which may result in unsuccessful self-organization. Hence, we use the better method, which can determine the size of the channel segment. We define a threshold for the minimum number of channels for each cell, λ. Then the number of channels assigned to each cell equals the following expression:

$$\min\ (\max\ (\lambda, |M|/|C|), |M|).$$



(a) cells and subcells (b) channel hoppings

Fig. 2. Example of self-organization

3) Segment Reuse: The channel segment needs to be reused under some circumstances. The constraint for reuse is to avoid two adjacent cells from being assigned with the same channel segment. Since each cell knows the cell numbers of adjacent cells, it is able to calculate which channel segments are assigned to the adjacent cells that have smaller cell indices than this cell. The reuse policy is to move to the next channel segment that is not the same as that of any adjacent cell with smaller indices.

B. Information Learning and Self-Organization within a Cell

After IHCs are selected, the next step is to have nodes in the same cell learn about each other's information, for cluster heads will be selected from each cell later. There are three problems to address. First one is how to determine the states (active/passive) of different nodes. Another problem is how an active node can efficiently guess the IHCs of passive nodes. The third problem is how to let nodes in the same cell learn the information of each other, with limited control message exchange. We will solve the three problems one by one.

1) Subcell Construction: To determine the state of each node, we apply the subcell concept here.

***Definition** 2:* A cell c_k is divided into a set of square subcells, S_k. Each subcell is in either active or passive state. Nodes in active subcells are active, and nodes in passive subcells are passive. Two adjacent subcells are in different states.

The size of each subcell is $l < L$, where L is the size of each cell. The value of l can be adjusted to ensure that the number of active nodes is similar to the number of passive nodes. Fig. 2(a) is an example of cells and subcells. In this example, the network is divided into four cells. Each cell is divided into four subcells. The subcells marked as "A" are active subcells, and the subcells marked as "P" are passive subcells. We let each node know the value of l and the initial state settings of subcells are based on geographical locations of the subcells. Therefore, it is able to calculate which subcell it is located in, and switches to the corresponding state.

2) Channel Hoppings: For an active node to guess the IHCs of passive nodes in the same cell, it first runs Algorithm 1 and gets the IHC of itself. Since they are in the same cell, their calculated IHCs are based on the same channel segment. When the IHCs of active nodes and passive nodes are different, active

nodes need to do channel hoppings until they reach the IHCs of passive nodes.

We define a channel hopping range for active nodes. Suppose node i is active and node j is passive. They are in the same cell, c_k. We define an IHC for i, denoted as IH_i. We set a hopping range $[IH_i - \Delta M, IH_i + \Delta' M]$ for i to scan, which means node i would scan ΔM channels down from IH_i and $\Delta' M$ channels up from IH_i in M_i. Later on, we will discuss how to analytically find ΔM and $\Delta' M$ so that the self-organization process is successful, i.e., every node learns the information of every other node in the same cell.

An example is shown in Fig. 2(b). Nodes i and j are in the same cell, with i as active and j as passive. The squares denote channels. Squares marked as grey are available channels. $IH(c_k)$ is the middle channel, as shown in Fig. 2(b). The ΔM and $\Delta' M$ here both equal to 1. After searching on $[IH_i - \Delta M, IH_i + \Delta' M]$, node i will find j's IH_j by receiving ACKs from j.

3) Information Exchange and Self-Organization: Active nodes send messages on the guessed IHCs of passive nodes. If a passive node receives the request from an active node, it would reply with an ACK. The transmission range used here is: $r_0 = \sqrt{2}L$. It ensures that a request from an active node can cover the whole cell. In our model, only two steps are needed for each node to know all other nodes' information in the same cell. Each node maintains two lists, L_a to store the list of active nodes, and L_p to store the list of passive nodes. The active node also maintains a list of channels with passive nodes, L_c. Initially, $L_a = \varnothing$, $L_p = \varnothing$, and $L_c = \varnothing$. The node information here contains the node ID and its location in the network. The procedure for information exchange and self-organization is:

1) Each active node scans the channel hopping range. For each channel, it sends a request message containing its information. Each passive node returns an ACK with its own node information to every request it receives on its IHC, and stores the active node's information in list L_a; Upon receiving the ACK, the active node stores the passive node information into list L_p, and stores the corresponding IHC of the passive node into L_c.
2) After 1), each active node switches back to each channel in L_c. The active node sends its passive nodes list L_p to the passive nodes in this channel. On the other hand, the passive node replies its list L_a to the active node. Note that the active nodes are time synchronized. If otherwise, each active node needs to wait until every active node completes the channel hopping.

C. Success Probability of Self-Organization

In this subsection, we analyze the *success probability of self-organization*, which is defined as the probability that a node in a cell successfully learns the ID and other information of another node in the same cell. In the self-organization procedure, when an active node scans through the channels to find passive nodes, it is still possible that the IHCs of some passive nodes are not accessible by this active node. From Section IV-B2, the channel searching range for an active node is $\Delta M + \Delta' M + 1$ channels. Among the $\Delta M + \Delta' M + 1$ channels, let m denote the number of channels which are available to the CRN communication. Let u_k and v_k denote the mean busy and idle durations on the kth channel of the $\Delta M + \Delta' M + 1$ channels. Let $X_k = 1$ or 0 denote if the kth channel is available or not. X_k follows Bernoulli distribution with parameter $\frac{v_k}{u_k + v_k}$. We assume that the PU activities on different channels are independent. If $v_k = v$ and $u_k = u$ for all k, then $m = \sum_{k=1}^{\Delta M + \Delta' M + 1} X_k$ follows the Binomial distribution with parameters $\frac{v}{u+v}$ and $\Delta M + \Delta' M + 1$. Otherwise, m approximately follows a normal distribution with mean $\sum_{k=1}^{\Delta M + \Delta' M + 1} \frac{v_k}{u_k + v_k}$ and variance $\sum_{k=1}^{\Delta M + \Delta' M + 1} \frac{u_k v_k}{u_k + v_k}$. Furthermore, among the m channels, not every channel is available to every node, due to spatial diversity of channel availability. Let p denote the probability that a channel among the m channels is available to a node. Next, we analyze the conditional success probability of self-organization, given m and p. The unconditional success probability can be computed utilizing the Binomial distribution, which m follows.

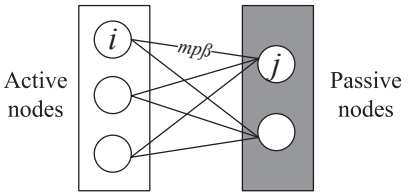


Fig. 3. The probability of connection as a bipartite graph

Lemma 1: [15] Let Z_{jk} denote the event that passive node j selects the kth channel in the range $[IH_j - \Delta M, IH_j + \Delta' M]$ as its home channel using Algorithm 3. Then the probability of Z_{jk}, denoted as β, is given as

$$\beta = \Pr(Z_{jk}) = \tfrac{1}{m}\left(1 - (1-p)^m\right). \tag{1}$$

Note that β depends on m and p only, neither j nor k.

Let node i be an active node and node j be a passive node. Let P_s denote the probability that node i can meet the passive node j in the m available channels. Let $\tilde{N}$ denote the number of passive nodes in the cell, and $\bar{N}$ denote the number of active nodes in the cell. We have the following theorem on the self-organization success probability.

Theorem 1: The P_s is lower bounded as follows,

$$P_s \geq 1 - (1 - m\beta p)^{\min(\tilde{N}, \bar{N})}. \tag{2}$$

Proof: The probability that i can meet j on kth ($1 \leq k \leq m$) channel among the m available channels is βp, i.e., if node j selects the kth channel as the home channel and the kth channel is available to node i. The total probability that node i meets node j in any of the m channels is

$$\sum\nolimits_{k=1}^{m} \beta p = m\beta p. \tag{3}$$

The $\tilde{N}$ passive nodes and the $\bar{N}$ active nodes in the cell of nodes i and j form a complete bipartite graph, with the active nodes on one side and passive nodes on the other side. Let

the cost of a link denote the *link connection probability*, which is the probability that two end nodes meet each other in the self-organization. By Eq. (3), the connection probability of every link is $m\beta p$, since (3) does not depend on i or j. The active node i and passive node j are disconnected only if all links of a cut are disconnected. Let n denote the number of links of the cut, then the probability that nodes i and j are disconnected due to this cut is $(1-m\beta p)^n$. We can see that this *nodes disconnection probability* increases when n decreases. Thus, the disconnection probability is maximized if the links of a mincut are disconnected. For a complete bipartite with $\tilde{N}$ passive nodes on the left and $\bar{N}$ active nodes on the right, the cardinality of the mincut between nodes i and j is $\min(d_i, d_j) = \min(\tilde{N}, \bar{N})$, where d_i is the degree of node i. Therefore, the disconnection probability, denoted as P_d, is upper bounded as

$$P_d \leq (1-m\beta p)^{\min(\tilde{N},\bar{N})}.$$

Accordingly, the connection probability between the active node i and passive node j through all possible routes in the bipartite graph is

$$P_s = 1 - P_d \geq 1 - (1-m\beta p)^{\min(\tilde{N},\bar{N})}.$$

■

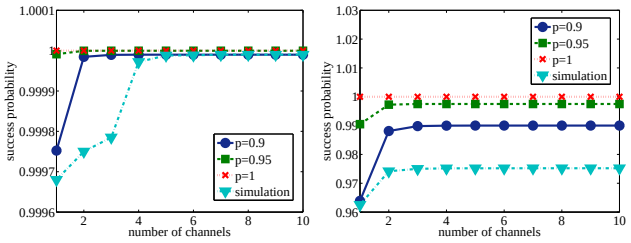
D. Estimation of Channel Hopping Range

Based on Theorem 1, we can find m that satisfies the self-organization success probability requirement, for given $p, \tilde{N}, \bar{N}$. From m, we can find ΔM and $\Delta' M$, the channel hopping range in the self-organization. Fig. 4 illustrates the self-organization success probability as a function of m. In the simulation, we generate primary users on each channel with random session requests. We can see that the analysis results match the simulation results very well. When $p = 0.9$ and there are only 3 passive nodes and 2 active nodes, $m = 3$ makes the success probability be as large as 0.99. If $\tilde{N}$ and $\bar{N}$ are larger, the success probability is even larger, e.g., equal to 0.99999 for $\tilde{N} = 5$ and $\bar{N} = 5$. Since the number of available channels follows Binomial distribution $B(j : \frac{v}{u+v}, \Delta M + \Delta' M + 1)$, as discussed in preceding subsection, then the channel hopping range $\Delta M + \Delta' M + 1 \approx m\frac{u+v}{v}$. For instance, if $\frac{v}{u+v} = 0.6$, which is a high spectrum utilization for primary users, then the channel hopping range $\Delta M + \Delta' M + 1 \approx 5$. Therefore, in the self-organization procedure, the active nodes only need to scan about 5 channels to ensure that all nodes will be found/connected (success probability close to 1).

V. Virtual Backbone Construction

A. Cluster Head Selection

The classical clustering approach [5] works as follows: (1) all nodes are initially uncovered; (2) an uncovered node i becomes a cluster head if it has the highest priority among its 1-hop uncovered neighbors including i; (3) the selected cluster heads and its connected 1-hop neighbors are marked as covered; repeat (2) and (3) on all uncovered nodes (if any).



(a) $\tilde{N} = 5, \bar{N} = 5$ (b) $\tilde{N} = 3, \bar{N} = 2$

Fig. 4. The self-organization success probability

In our model, the cluster heads are selected distributively in each cell using the above approach. Based on our assumption, the transmission range of each node can be adjusted. Here, we set the transmission range for cluster head selection as: $r_1 = \frac{2\sqrt{2}}{3}L = \frac{1}{3}R$.

We can improve the head selection efficiency by utilizing the information collected by nodes in the same cell. Since each node learns about all other nodes' ID and location in the same cell from Section IV, it can run the classical clustering approach by itself, without exchanging information with other nodes. The priority we use here is the node's ID. The node with the lowest ID value has the highest priority. For example, in Fig. 5, there are three nodes, i_1, j_1, and j_2, in the upper left cell. Since the three nodes already know each other's ID and location in one cell, they also know the nodes' ID and locations within range r_1 in one cell ($r_1 < L$). Therefore, they do not need to exchange any control message. All three nodes would choose i as the cluster head after applying the classical clustering approach distributedly.

Since our cluster heads are selected based on each cell, they may be different from the results if running the clustering approach in the whole network without any cell. We need to prove that the coverage is unchanged in our algorithm. We use H to denote the set of cluster heads.

***Theorem** 2:* The coverage remains unchanged if the cluster heads are selected based on cells.

Proof: For a node i, after the cluster heads are selected in each cell, i must be covered. This is because, based on our cell division, i must belong to a cell c_k. Then in c_k, there exists a node h, $h \in H$, ($h = i$ or $h \neq i$), that is a cluster head in c_k and connected to i. Thus, i is covered. ■

B. Backbone Node Selection

After cluster heads are selected from cells, the next step is to select the backbone nodes from cluster heads. Here, we apply the approach in [5], consisting of *marking process*: Each node is marked if it has two unconnected neighbors. Otherwise, it is unmarked; and *pruning rule*: A marked node can unmark itself if its neighbor set is covered by a set of connected nodes with higher priorities.

The approach is to reduce the number of cluster heads and construct the backbone while ensuring the connectivity and coverage. The marked cluster heads are the backbone nodes. Details are in [5]. The priority among nodes still depends on their ID. The remaining problem is how the cluster heads

Algorithm 2 h sends requests to cluster heads in an adjacent cell c'_k

1. $Temp = M_h$
2. **while** $Temp \neq \emptyset$ **do**
3. $\quad temp = Pro(h, M_h, c_{k'})$
4. $\quad h$ broadcasts a request message using $temp$
5. $\quad$ Remove $temp$ from $Temp$
6. **end while**

exchange information with each other in order to run the above marking process and pruning rule.

1) Cluster Head Communication: A cluster head needs to exchange information with all cluster heads around it. Therefore, it would exchange information within its adjacent eight cells. The transmission range used here is: $r_2 = R$, which is the same as the data transmission range R of each node. Since $r_2 = R = 2\sqrt{2}L$, it ensures that each cluster head is able to reach the adjacent eight cells, which covers all the cluster heads within r_2 around it. For example, node i_5 in Fig. 5 is able to cover the upper left cell using r_2.

Since nodes in different cells use different cell IDs to calculate their IHCs, cluster heads need to guess the IHCs used by others in adjacent cells. For a $h \in H$ that is located in cell c_k, it is not hard for h to guess the IHC of another cluster head in H, which is located in the cell $c_{k'}$ adjacent to c_k. The process for node h to send the request message to other cluster heads in cell $c_{k'}$ is in Algorithm 2.

2) Efficiency Improvement: Since cluster heads are also nodes in the network, they have active/passive states. A cluster head sends a request message only when it is active. Cluster heads in the same cell know each other's information (ID, location). To increase efficiency, the request message sent by each active cluster head would include the information of all the cluster heads in the same cell. Also, if a passive cluster head receives a request message from its IHC, it replies a message with the information of all cluster heads in its cell. If a cluster head receives a request or reply message from an adjacent cell, it would mark that cell as known. If no message is received through the above process, the active cluster head would backoff and retry again later until receiving the replying or request message. The whole procedure continues for each cluster head until the adjacent cells are all marked as known.

For example, in Fig. 5, node $i_1 \in H$ is an active cluster head and wants to learn the cluster heads in the bottom left cell. i_2 and i_3 are passive cluster heads in the bottom left cell. If $IH_{i_2} \notin M_i$, then i_1 cannot reach i_2 directly. However, if i_1 can reach i_3, it would receive a reply message from i_3. i_3 would send both i_2 and i_3's information to i_1. i_1 would mark this cell as known. i_3 also marks the upper left cell as known, based on the information in the request sent by i_1. Then i_3 shares the information with i_2. Therefore, i_2 also marks the upper left cell as known.

3) Backbone Formation: After cluster heads in H learn about the neighbor information in adjacent cells, they perform the marking process and pruning rule, and then form the final

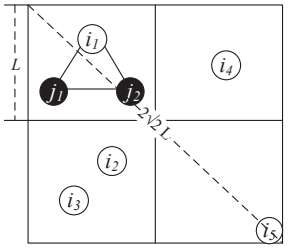


Fig. 5. Example of backbone nodes selection.

virtual backbone. Now we need to prove that the connectivity and coverage is unchanged through constructing the backbone using our approach. We have the following theorem:

***Theorem** 3:* For any nodes i and j, if they are connected in the original network, they are both covered and still connected through nodes in the backbone node set B.

Proof: In [5], the authors proved that their approach ensures the connectivity and coverage. What we need to prove here is that the backbone nodes B' selected in [5] are still connected using B. The difference is that their marking process and pruning rule is performed for all cluster heads within r_2, and ours is performed for all cluster heads in adjacent cells. For a cluster head h_1, it conducts the marking process and pruning rule for adjacent cells. Cluster heads outside the adjacent cells of h, but within the range of r_2, are connected because none of them is removed. For cluster heads within the adjacent cells of h, they are connected since the results are the same as [5]'s approach, which ensures the connectivity and the coverage using the same broadcast process in [5]. ■

VI. End-to-End Data Transmission

We first describe how each node chooses the data transmitting channel and how the single-hop links are built. Then, we propose a scheme of having a virtual backbone to calculate the area route, and using multiple links to transmit data simultaneously intra and inter areas until the destination node is reached.

A. Single Hop Transmission

To build single-hop links for the network, each node uses the active/passive states. The home channel for passive nodes here cannot be the same as the IHCs in the previous section. This is to allow multiple links transmitting simultaneously. If the active nodes transmit through the IHCs of the passive nodes, the interference would be relatively high among links nearby. This is because IHCs of nodes in the same cell, which are geographically close, are selected from the same channel segment. Therefore, we need to choose a new home channel for every node to transmit, which is called *transmission home channel* (THC).

1) THC Selection: We adopt the approach in [11] for the nodes to choose THCs and guess the THCs of others, which has minimal control overhead and is highly effective. Algorithm 3 describes how to select the THC of node i with the available channel set M_i, denoted as $Pro2(i, M_i)$. Note that Algorithm 3 is similar to Algorithm 1. The difference

Algorithm 3 $Pro2(i, M_i)$, to compute THC for node i

1. Set i as the seed for the pseudo-random number generator Z.
2. Let $\mathcal{Q} = M$ // The total channel set
3. **repeat**
4. $k = Z(|\mathcal{Q}|)$ // Generate k such that $1 \leq k \leq |\mathcal{Q}|$
5. $q = \mathcal{Q}(k)$ // $\mathcal{Q}(k)$ is kth channel in $\mathcal{Q}$
6. $\mathcal{Q} = \mathcal{Q} \backslash \{q\}$ // Remove q from $\mathcal{Q}$
7. **until** $q \in M_i$
8. Return q // Selected THC

here is the value of $\mathcal{Q}$. Here, $\mathcal{Q}$ equals M, instead of the channel segment assigned to the corresponding cell. Also, the algorithm has been deliberately designed to make the channel estimation (to be discussed) highly successful. It has a subtle difference from a naive random channel selection that simply picks a channel from set M_i at random. This subtle difference has a profound impact on the success probability for a node to estimate the home channel of another node. This has been proven in [11].

2) THC Estimation: When an active node i wants to transmit packets to a passive node j, it estimates the THC of node i as $Pro2(j, M_i)$, i.e., using node j's ID, but node i's accessible channel set M_i as parameters. Then node i switches to channel $Pro2(j, M_i)$. If the intended receiver, node j, is in fact on the estimated channel $Pro2(j, M_i)$, then the rendezvous is successful and the packet transmission starts. Algorithm 3 has been designed to ensure that the successful probability of the channel estimation, i.e., $\Pr(Pro2(j, M_i) = Pro2(j, M_j))$, is high. This is also proven in [11]. This approach does not require any exchange of control messages, which significantly streamlines the communication and reduces overhead.

3) State Sequence Generation: After each node has its THC selected, we need to decide the active/passive state sequences over a time period. There is a potential issue: when node i switches to the THC of node j to send packets, node j itself has switched to the home channel of another node. To resolve this issue, we adopt an approach similar to [15] and consider two *variants*, depending on if a node knows the number of nodes in its single-hop neighborhood. Since each node in our network is equipped with a GPS device, the cognitive radio is programmed to have the GPS waveform so that each node can receive the GPS signal to have a common time reference, i.e., all nodes are time synchronized while operating in time slotted mode.

Variant 1: The node does not know the number of nodes in its single-hop neighborhood. This happens when the number of nodes in the neighborhood changes dynamically, and we do not want to have the overhead or incur a control structure to keep track of the number of nodes at each node. In this case, we let node i use a function $g(i, t)$ to compute its state in time slot t. The function $g(\bullet)$ is a pseudo-random number generator that uses i and t as the seed to generate a random number of either 0 or 1. If $g(i, t) = 1$, then node i is a passive node in slot t and stays on its home channel in this slot. If $g(i, t) = 0$, node i is an active node in slot t. It selects a passive node and switches to the home channel of the selected passive node for packet transmission. Note that, node i knows whether another node, say node j, is a passive node, by calling the function $g(j, t)$ to find the status of node j.

Variant 2: Each node knows the number of nodes in its single-hop neighborhood. If we know $n = |N|$, which is the number of nodes in the network, then we can generate the node status, such that the number of passive nodes and the number of active nodes are balanced, to maximize throughput. Specifically, in time slot t, every node uses a pseudo-random number generator function $g'(t, n)$ that uses t as the seed to generate the states for all n nodes, denoted as $[g_1, g_2, \ldots, g_n]$ with $g_i = 1$ or 0, denoting that node i is a passive or an active node. To balance the number of passive nodes and the number of active nodes, we let the pseudo-random number generator continue to generate $[g_1, g_2, \ldots, g_n]$ until $\sum_{i=1}^{n} g_i = \lceil \frac{n}{2} \rceil$, i.e., the number of passive nodes is $\lceil \frac{n}{2} \rceil$. Note that as every node uses the same seed t, every node would need exactly the same number of rounds of the pseudo-random number generator to get $\sum_{i=1}^{n} g_i = \lceil \frac{n}{2} \rceil$. Hence the $[g_1, g_2, \ldots, g_n]$ generated by each node is still the same. With the number of passive nodes being equal to $\lceil \frac{n}{2} \rceil$, the spreading of both passive and active nodes into different channels is maximized, which maximizes the number of simultaneous transmissions (on different channels) and, hence, the throughput.

B. Multihop Transmission

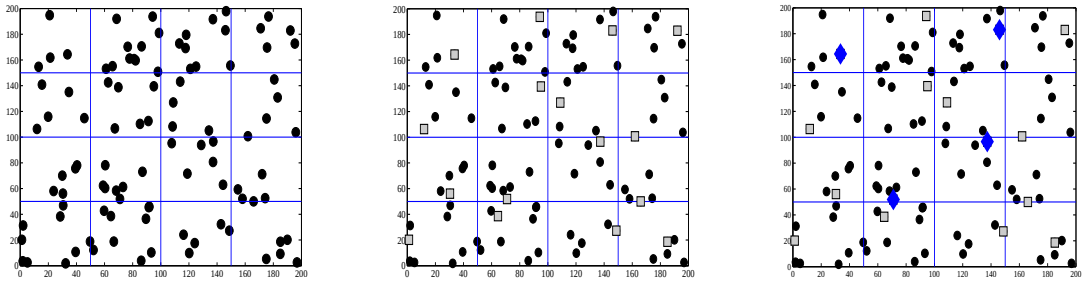
After the single-hop links are built, to implement the end-to-end data transmission, the source node needs to know the route to reach the destination. If the source node calculates a route of individual nodes, the overhead and delay would be very high. With the virtual backbone structure, we can provide an efficient multihop transmission scheme.

1) Area route: Since the virtual backbone is connected and covers the whole network, the backbone node that the source node is attached to can calculate an *area route* from the source area to the destination area.

***Definition** 3:* The area route is a set of areas that the source node needs to pass through to reach the destination node. Each area is a set of a backbone node and all nodes attached to it.

As shown in Fig. 1, the area route from source S to destination D is $A_1 \rightarrow A_2 \rightarrow ... \rightarrow A_k$. Source S belongs to A_1 and destination D belongs to A_k. The backbone node only needs to communicate with other backbone nodes until reaching the backbone node that the destination node is attached to. The overhead for a backbone node to calculate the area route is much less, compared to the one that the source node calculates the full path to reach the destination. We can apply any classical routing algorithm among backbone nodes to calculate the area route.

2) Intra-area Data Transmission: For data transmission within a single area, since each node knows the number of nodes in the area, it applies the Variant 2 in the preceding subsection to generate its state: 1) based on the border nodes



(a) nodes and cells (b) cluster head selection (c) backbone node selection

Fig. 6. Process of a backbone construction.

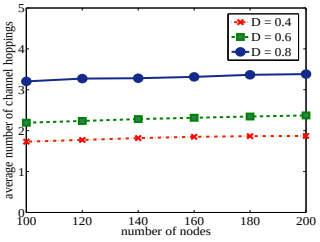


Fig. 7. # of hoppings *VS* nodes

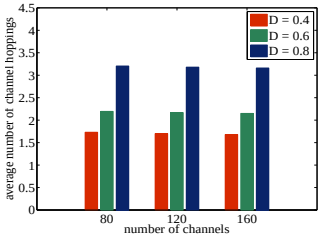


Fig. 8. # of hoppings *VS* channels

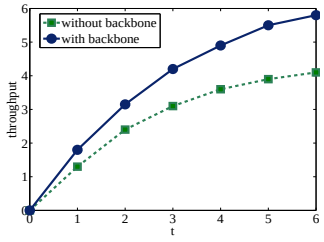


Fig. 9. Throughput over time

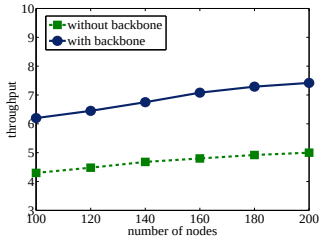


Fig. 10. Throughput *VS* # of nodes

schedule, each node knows the number of nodes in this area in the current time slot. Let N' denote this number; 2) each node generates the node status $[g_1, g(2), \ldots, g_{N'}]$; 3) if $g_i = 0$, then node i is active, and otherwise, node i is passive, and selects its THC to keep listening; 4) the active node selects a passive node, e.g., based on the first packet of the packet queue, and switches to the estimated THC of the passive node.

3) Inter-area Data Transmission: For data transmission between different areas, we need to design an active schedule for different areas. We make use of the *gateway nodes* here. If a node is at the border of two or more areas, then it is a gateway node. To select the gateway nodes, we can set a threshold of the distance differences from a node to two backbone nodes. If the distance difference is within the threshold, then the node becomes a gateway node. This can be easily implemented, since each node has the GPS device and knows the location of itself and the backbone nodes.

A gateway node needs to communicate with all nodes of each area it belongs to. Except the gateway nodes, a node communicates with the neighbors in the same area only, even though it can reach nodes in another area. A gateway node sequentially joins the areas it belongs to. Let $\{A_1, A_2, \ldots, A_k\}$ denote the set of the areas that a gateway node, i, belongs to. At time slot t, i would join area A_{k_1} where $k_1 = (t \bmod k)$ +1. When i joins area A_{k_1} $(1 \leq k_1 \leq k)$, the nodes in area A_{k_2} $(k_2 \neq k_1,\ 1 \leq k_2 \leq k)$ know that node i is not in area A_{k_2}, by the control information from the backbone node, and hence do not try to communicate with node i. Nodes from an area would send the data to a gateway node when it joins this area. Then after the gateway node joins another area, it would forward the information received in the previous area to the nodes in the current area.

The use of gateway nodes can improve the performance in two aspects. First, the throughput is increased. This is because there can be several gateway nodes in one area, and hence, there can be simultaneous data transmission on different channels between multiple nodes and the gateway nodes, and between gateway nodes and gateway nodes. Second, the backbone nodes only need to calculate area routes, and there are no data packets being forwarded through the backbone nodes. Hence the traffic loads on the backbone nodes are minimized, which avoids the congestion at the backbone nodes if the traffic of all nodes have to go through the backbone nodes. The packet delay is reduced as well.

VII. Simulation

A. Simulation Settings

We distribute nodes in a 200×200 unit square. The cell size is set as 50×50. We also generate 40 primary users, which are randomly active and occupy some channels. We vary the following two network parameters.

1) number of nodes: $100 \sim 200$ with an increment of 20;
2) number of channels: $80 \sim 160$ with an increment of 40.

We compare the throughput and delay with the approach in [11], which does not have a virtual backbone construction.

B. Simulation Results

We first show the cost of self-organization in terms of the average number of channel hoppings for an active node to reach a passive node in Figs. 7 and 8. We compare the average number of channel hoppings with three different channel busy time ratios, i.e., $\frac{u}{u+v} = 0.4$, 0.6, and 0.8. The results show that the number of channel hoppings for all cases are less than 5, which verifies the analysis in Section IV-D. Fig. 7 shows that when the number of nodes increases, the number of channel hoppings increases slowly. Fig. 8 shows that the increase of the channel availability does not have a huge influence on the number of channel hoppings.

Next, we set the number of nodes as 100 and the number of total channels as 80. Also, we show the process of backbone

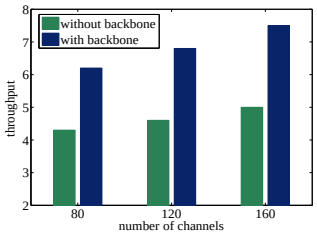


Fig. 11. Throughput *VS* channels

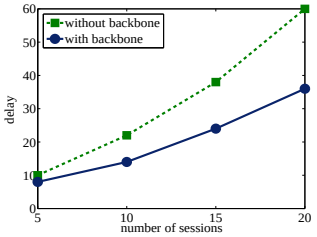


Fig. 12. Delay *VS* session #

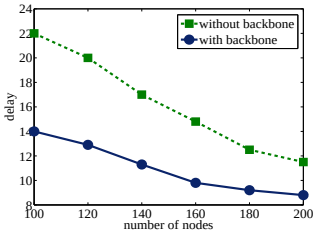


Fig. 13. Delay *VS* # of nodes

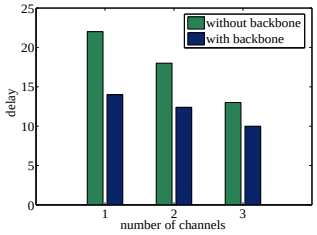


Fig. 14. Delay *VS* # of channels

node selection using our model. The process is shown in Fig. 6. Fig. 6(a) shows the nodes distribution in the network with cell divisions. In Fig. 6(b) the square nodes are the selected cluster heads from each cell. Fig. 6(c) shows the selected backbone nodes (the diamond ones).

Moreover, we implement the algorithm in [11], which does not have a virtual backbone construction, and compare it with our approach. In the network without a virtual backbone, each node does not know the number of nodes in the network, and uses the Variant 1 in Section VI to reach their next hop. Also, since no virtual backbone exists, the source needs to calculate the route to the destination itself.

The comparison results of throughput are shown in Figs. 9, 10, and 11. The throughput is the average throughput for all the active sessions in the network. In Fig. 9, the throughput at the first 6 time slots are shown. The one with the virtual backbone structure increases faster than the one without the virtual backbone over time. At the 6th time slot, our approach achieves almost 1.6 times over the one without virtual backbone. In Fig. 10, the results show that the throughputs of both approaches increase slowly when the number of nodes increases. In Fig. 11, the throughputs of both approaches increase while the number of channels increases. Overall, the throughput with the virtual backbone structure is much larger than the one without a virtual backbone structure.

Finally, we compare the packet delay between the two approaches. The results are shown in Figs. 12, 13, and 14. In Fig. 12, we vary the number of active sessions, while setting the number of nodes as 100. The delay in both approaches increases, when the number of active sessions increases. The delay for the approach without the virtual backbone structure increases more rapidly. In Fig. 13, the delay in both approaches decreases, when the number of nodes increases. Nevertheless, the speed of delay decrement becomes slower as the number of nodes increases. This is because when there are more nodes, the average number of channels available to each node is smaller. In Fig. 14, we vary the number of total channels in the network, while setting the number of nodes as 100. The results show that when the number of total channels increases, the delay decreases for both approaches. Overall, the delay of the approach with the virtual backbone structure is less than the one without a virtual backbone structure.

VIII. Conclusion

In this paper, we present a comprehensive approach on self-organization, virtual backbone construction, and end-to-end data transmission for CRNs, without relying on a common control channel (CCC). Each node makes use of the location information and adjustable transmission range for the virtual backbone construction. We let each node choose its own channel for data transmission and reduce the interference among different links. We propose an efficient scheme for end-to-end data transmission. The simulation results verify our theoretical analysis and show that the efficiency of our approach is significantly improved compared with the one without a virtual backbone.

References

[1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Computer Networks*, 2006.

[2] B. Liang and Z. Haas, "Virtual backbone generation and maintenance in ad hoc network mobility management," in *Proc. of IEEE Infocom*, 2000.

[3] K. Alzoubi, P. Wan, and O. Frieder, "Message-optimal connected dominating sets in mobile ad hoc networks," in *Proc. of ACM MobiHoc*, 2002.

[4] D. Dubhashi, A. P. A. Mei, J. Radhakrishnan, and A. Srinivasan, "Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons," in *Proc. of ACM-SIAM Symp. Discrete Algorithms*, 2003.

[5] J. Wu and F. Dai, "Virtual backbone construction in manets using adjustable transmission ranges," *IEEE Transactions on Mobile Ccomputing*, 2006.

[6] J. Wu and F. Dai, "Efficient broadcasting with guaranteed coverage in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, 2005.

[7] L. Le and E. Hossain, "Osa-mac: A mac protocol for opportunistic spectrum access in cognitive radio networks," in *Proc. of IEEE WCNC*, 2008.

[8] M. Pan, C. Zhang, P. Li, and Y. Fang, "Joint routing and link scheduling for cognitive radio networks under uncertain spectrum supply," in *Proc. of IEEE Infocom*, 2011.

[9] Y. Yuan, P. Bahl, R. Chandra, P. Chou, J. Ferrell, T. Moscibroda, S. Narlanka, and Y. Wu, "Knows: Cognitive radio networks over white spaces," in *Proc. of IEEE DySpan*, 2007.

[10] L. DaSilva and I. Guerreiro, "Sequence-based rendezvous for dynamic spectrum access," in *Proc. of IEEE DySpan*, 2008.

[11] C. Xin, M. Song, L. Ma, S. Shetty, and C.-C. Shen, "Control-free dynamic spectrum access for cognitive radio networks," in *Proc. of IEEE ICC*, 2010.

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

[13] Z. Lin, H. Liu, X. Chu, and Y.-W. Leung, "Jump-stay based channel-hopping algorithm with guaranteed rendezvous for cognitive radio networks," in *Proc. of IEEE Infocom*, 2011.

[14] Y. Zhang, Q. Li, G. Yu, and B. Wang, "Etch: Efficient channel hopping for communication rendezvous in dynamic spectrum access networks," in *Proc. of IEEE Infocom*, 2011.

[15] C. Xin, M. Song, L. Ma, and C.-C. Shen, "Performance analysis of a control-free dynamic spectrum access scheme," *IEEE Transactions on Wireless Communications*, 2011.