

SPECTRUM MANAGEMENT AND CROSS-LAYER PROTOCOL DESIGN IN COGNITIVE RADIO NETWORKS

A Dissertation
Submitted to
the Temple University Graduate Board

in Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

by
Ying Dai
December, 2014

Examining Committee Members:

Dr. Jie Wu, Department of Computer and Information Sciences (Advisor)
Dr. Chiu C. Tan, Department of Computer and Information Sciences (Committee chair)
Dr. Li Bai, Department of Electrical and Computer Engineering
Dr. Wei-Shih Yang, Department of Mathematics (External reader)

ABSTRACT

Spectrum Management and Cross-layer Protocol Design in Cognitive Radio Networks

by

Ying Dai

Cognitive radio networks (CRNs) are a promising solution to the channel (spectrum) congestion problem. This dissertation presents work on the two main issues in CRNs: spectrum management and cross-layer protocol design.

The objective of spectrum management is to enable the efficient usage of spectrum resources in CRNs, which protects primary users' activities and ensures the effective spectrum sharing among nodes. We consider to improve the spectrum sensing efficiency and accuracy, so that the spectrum sensing cost is reduced. We consider the pre-phase of spectrum sensing and provide structures for sensing assistance. Besides the spectrum sensing phase, the sharing of spectrum, or the channel allocation, among nodes is also the main component in the spectrum management. We provide our approach to achieve a reliable and effective channel assignment.

The channel availabilities for different nodes in CRNs are dynamic and inconsistent. This poses challenges on the MAC layer protocols for CRNs. Moreover, due to the lack of knowledge on primary users, they can suddenly become available during the secondary users' data transmission. Therefore, for a end-to-end data transmission in CRNs, the routing algorithm is different from the existing routing algorithms in traditional networks. We consider the cross-layer protocol design, and propose the solutions for efficient data transmission. We propose the novel routing protocol design considering the boundaries of PUs. Also, an effective structure for reliable end-to-end data transmission is presented, which makes use of the area routing protocol. We build a USRP/Gnuradio testbed for the performance evaluation of our protocols.

Keywords: Cognitive radio networks, spectrum management, cross-layer protocol, performance evaluation.

To my parents

ACKNOWLEDGEMENTS

This research could not have been finished without the guidance of my committee members, support from my family, and help of many friends.

I would like to express my deepest gratitude to my advisor, Dr. Jie Wu, for his immeasurable guidance, caring, and patience. He has taught me how to think about research problems, provided me with an excellent environment for doing research, and shown me the importance of constant hardworking to approach any achievement. He has always been a strong advocate and a role model for me. There are so many things that I learned from him, which will be valuable assets for my future.

I would like to thank Dr. Chiu C. Tan, Dr. Li Bai, and Dr. Wei-Shih Yang, for being my committee members. I sincerely appreciate their help and their valuable comments to complete my dissertation.

Special thanks go to my lab-mates, for the fun we had together and the support we gave each other. I will easily miss hanging out with them all. To the staff of CIS department, thanks for their help and bearing with me.

Finally, my most heartfelt thanks to my parents, Ping Dai and Chunlan Zhang for their unconditional love. Thanks for supporting me, encouraging me, and always believing in me.

TABLE OF CONTENTS

ABSTRACT	ii
DEDICATION	iii
ACKNOWLEDGEMENTS	iv
LIST OF FIGURES	viii
LIST OF TABLES	xii
CHAPTER	
1. INTRODUCTION	1
1.1 Background	1
1.2 Major Challenges	2
1.3 Contributions	3
2. LITERATURE REVIEW	5
2.1 Cooperation For Spectrum Sensing	5
2.2 Channel Assignment	6
2.3 Routing Protocols and Route Maintenance	7
2.3.1 Route Selections	7
2.3.2 Joint Routing Protocols	8
2.3.3 Route Maintenance	8
2.3.4 Architecture Design in CRNs	9
3. SPECTRUM SENSING IMPROVEMENT IN CRNS	11
3.1 Pre-phase for Spectrum Sensing	11
3.1.1 Preliminaries	11
3.1.2 Construct State Transition Diagram	13
3.1.3 Sense-in-order Model	14
3.1.4 Sort Sensing Order	17
3.1.5 Extended Model for SIO	19
3.1.6 Simulations	21
3.1.7 Conclusion	24
3.2 Hitchhiking for Sensing	25

3.2.1	System Model	25
3.2.2	Core-Only Structure	26
3.2.3	Spectrum Sensing with Cores	29
3.2.4	Core Evolution	33
3.2.5	Mixed Cluster-core Structure	35
3.2.6	Simulations	38
3.2.7	Conclusion	42
3.3	Chapter Conclusion	42
4.	CHANNEL ASSIGNMENT IN CRNS	44
4.1	Effective Channel Assignments	44
4.1.1	Preliminaries	45
4.1.2	Basic Local Algorithms	46
4.1.3	Advanced Local Algorithm	48
4.1.4	Improved Conflict Resolutions	53
4.1.5	Simulations	55
4.1.6	Conclusion	60
4.2	Channel Assignment Under Dynamic Source Routing	61
4.2.1	Preliminaries	61
4.2.2	Problem Formulation	63
4.2.3	Single Route Channel Assignment	64
4.2.4	Multi-Route Channel Assignment	73
4.2.5	Simulations	77
4.2.6	Conclusion	80
4.3	Chapter Conclusion	80
5.	SPECTRUM AWARE ROUTING IN CRNS	82
5.1	Routing Protocol with Boundary Nodes	82
5.1.1	Preliminaries	83
5.1.2	Problem Formulation	84
5.1.3	Routing Protocol with Directional Antenna	87
5.1.4	Feasibility Improvement	98
5.1.5	Extensions	102
5.1.6	Simulations	104
5.1.7	Conclusion	109
5.2	Forwarding Node Set Selection	110
5.2.1	Problem Formulation	112
5.2.2	FN Set Selection Model	114
5.2.3	Performance Analysis	123
5.2.4	FN Adjustment	124
5.2.5	Simulations	126
5.2.6	Conclusion	128
5.3	Chapter Conclusion	129

6. INFRASTRUCTURE DESIGN FOR CRNS	130
6.1 Virtual Backbone Construction	130
6.1.1 Preliminaries	131
6.1.2 Problem Formulation	132
6.1.3 Self-Organization	133
6.1.4 Backbone Nodes Selection	140
6.1.5 End-to-End Data Transmission	144
6.1.6 Simulations	149
6.2 Chapter Conclusion	153
7. CONCLUSION	154
7.1 Summary of Contributions	154
7.2 Future Research	155
BIBLIOGRAPHY	156

LIST OF FIGURES

Figure

1.1	An example of a CRN with three PUs and four SUs.	2
3.1	Sense-in-order can reduce delay and save energy.	14
3.2	State transition diagram.	16
3.3	Directional antenna creates more channel availabilities.	19
3.4	Comparison of success percentage when varying network settings.	23
3.5	Study of parameter influences	23
3.6	Comparison of success percentage when varying algorithm settings.	24
3.7	An example of constructing core structure.	29
3.8	The interaction example between u and v ($c_u = v$).	31
3.9	Cluster head selection from cores.	38
3.10	Comparison of core-only scheme with different size constraints and random sensing scheme.	40
3.11	Comparison of core-only scheme under different information exchange frequencies.	40
3.12	Construction and performance comparison of cluster-core scheme.	41
4.1	The example topology for channel assignment.	46
4.2	The channel assignment process of the star charged by node c	54
4.3	Comparison of delivery rate in two models.	56
4.4	Comparison of delivery rate among three proposed algorithms.	57
4.5	Comparison of assigned link rate among three proposed algorithms.	57

4.6	Comparison of rounds among three proposed algorithms.	58
4.7	Comparison of assigned link rate among node-link-based, greedy, and optimal algorithms.	59
4.8	Comparison of rounds among node-link-based, greedy, and optimal algorithms.	59
4.9	Comparison among three conflict resolution strategies.	60
4.10	An example of single route.	67
4.11	Virtual nodes of a single route.	68
4.12	Multi-route with alternative nodes.	74
4.13	Single route.	78
4.14	Preferences.	78
4.15	Modified preferences.	78
4.16	U VS nodes.	78
4.17	U VS $Chs.$	78
4.18	Base nodes.	78
4.19	G VS nodes.	79
4.20	G VS $Chs.$	79
4.21	U VS nodes.	79
5.1	Testbed for showing the characteristic of a boundary node.	89
5.2	Receiving results at sector I	90
5.3	Receiving results at sector II	90
5.4	Two possible routes from S to D	91
5.5	Four cases of link ab located in one or multiple PU areas, detected by boundary nodes.	95

5.6	An example of missing boundary detection for the PU area of m . . .	99
5.7	Hop number.	106
5.8	Delay.	106
5.9	Varying PUs.	106
5.10	Comparison of the total delay by varying network parameters. . . .	108
5.11	Comparison of the average route length by varying network parameters.	108
5.12	Comparison of models with and without virtual boundary nodes. . .	109
5.13	The FN adjustment is needed for x , rather than u	113
5.14	An example of the modules for node u	121
5.15	Comparison of delay under different network environment parameters.	126
5.16	Comparison of FN adjustments under different network environment parameters.	127
6.1	Example of the end-to-end transmission using virtual backbone. . .	132
6.2	Example of self-organization	135
6.3	The probability of connection as a bipartite graph	138
6.4	The self-organization success probability	141
6.5	Example of backbone nodes selection.	141
6.6	Number of hoppings VS nodes.	150
6.7	Number of hoppings VS channels.	150
6.8	Process of a backbone construction.	150
6.9	Throughput over time.	151
6.10	Throughput VS nodes.	151
6.11	Throughput VS channels.	152

6.12	Delay <i>VS</i> session number.	152
6.13	Delay <i>VS</i> number of nodes	152
6.14	Delay <i>VS</i> number of channels.	152

LIST OF TABLES

Table

3.1	Simulation settings for SIO evaluation.	21
3.2	Simulation settings for hitchhiking evaluation.	39
4.1	Admissible channel set on each link.	53
4.2	Conflict probability of every channel on each link.	53
4.3	Weight of every channel on each link.	53
4.4	Simulation settings for local channel assignment algorithms.	55
4.5	Simulation settings for channel assignment under DSR.	77
5.1	An example of weighted route length.	97
5.2	Simulation settings for boundary-based routing.	104

CHAPTER 1

INTRODUCTION

Cognitive radio networks (CRNs) are a promising solution to the channel (spectrum) congestion problem. However, the high channel dynamics pose a lot of challenges to different layers in CRNs. The focus of this dissertation is the spectrum management and the other one is cross-layer protocol design, which are more challenging compared to traditional wireless networks.

1.1 Background

Today we are facing a dilemma of rapidly increasing demand of wideband wireless access, and shrinking out of unallocated spectrum. Studies indicate that, at any given time and location, there exists a large portion of under-utilized licensed spectrum [1]. Thus, people are exploiting new ways of transmitting on licensed bands when these bands are not fully used, which are also referred as channels.

CRNs are the key technology that enables next generation communication networks [2], also known as dynamic spectrum access (DSA) networks. There are two types of users in CRNs: primary users (PUs) and secondary users (SUs). PUs are privileged users and their transmission cannot be interfered by SUs, which are also referred as nodes in CRNs. Cognitive radio technology allows the SUs to utilize the spectrum more efficiently in an opportunistic fashion, without interfering with the PUs. As shown in Fig. 1.1, the SUs $N1$, $N2$, $N3$, and $N4$, are able to make opportunistic use of channels for transmission without disturbing PUs $PU1$, $PU2$, and $PU3$.

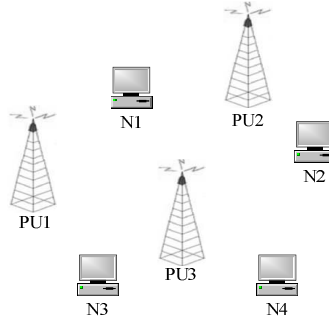


Figure 1.1: An example of a CRN with three PUs and four SUs.

CRNs enable each node (SU) to perform these functions: spectrum sensing, channel management, and spectrum mobility. Spectrum sensing is the physical layer technique, which makes use of the energy detection, signal characteristic analysis, and so on, to distinguish the PUs' transmission from SUs' transmission. Channel management is related to how to maximize the channel utilization among SUs. It focuses on the single link transmission of SUs while protecting the PUs. Spectrum mobility considers the situation when the PUs suddenly become active during the SU sessions, which result the broken of SU links.

1.2 Major Challenges

The major challenges of our work lie in different aspects, according the various phases of CRNs.

The first challenge is in the spectrum sensing phase [3]. To protect the PUs in CRNs, one important phase is the spectrum sensing. Before a node transmits data, it needs to perform spectrum sensing and make sure the channel (spectrum) it selects is not occupied by PUs. Most of the work considers the spectrum sensing phase as well as the data transmission phase after it. However, in CRNs with multiple channels, one aspect that may be neglected is the phase before spectrum sensing happens. That is, choosing which channel to sense first can reduce the total number of channels to sense before finding a channel that is not occupied by PUs. Therefore, it is potentially

very beneficial if the node can know which channels have higher probabilities available before sensing them.

The second challenge is the channel assignment among nodes in CRNs [4]. The fundamental difference between CRNs and traditional wireless networks is that the available channel sets are dynamic, and their availabilities vary over time and space. The channel assignment problem in CRNs is more complicated considering the existence of PUs and dynamic channel availabilities. Here, the channel assignment refers to as how to assign the available channels so that certain optimization objectives, such as throughput, spectrum efficiency, the number of nodes served, and fairness can be achieved.

The third challenge is the routing issues in CRNs [5]. Since nodes in CRNs need to quit immediately when PUs become active and occupy a channel, some links can possibly be suddenly broken. The dynamics of channel availabilities result in the difficulty of routing in CRNs, and in carrying out end-to-end data transmission. The initial route selection is very important, which needs to consider the influences of PU activities into consideration. In addition, the route reliability and route recovery cannot be neglected considering the inevitable route broken issues during transmission.

The fourth challenge is the architecture design in CRNs. To make CRNs more practical and efficient, the architecture design problem becomes interesting and important. A well-designed architecture in CRNs can bring a lot of benefits in different layers to CRNs. However, it is non-trivial for nodes to exchange their information and form into a reliable structure. The MAC and routing protocols based on the new architecture need to be studied as well.

1.3 Contributions

From the above discussions regarding the major challenges, the contributions of this dissertation fall in the fields of spectrum management and cross-layer protocol

design, which can be summarized as follows:

- We consider the phase before spectrum sensing happens, which is how to select channels for sensing in CRNs. We aim at reducing the total number of channels that a node needs to sense before finding an available one. The corresponding structure and sharing scheme is proposed.
- We design a fast convergent-localized protocol that assigns conflict-free channels, so as to maximize connectivity in multihop CRNs, which considers the high dynamics of channel availability and the low time-consuming requirement.
- We study the routing problem by first construct the more reliable route during the route selection phase. Moreover, through the efficient relay node selection, the reliability of the end-to-end transmission is improved. When the inevitable broken of the established route happens, our distributed rerouting scheme is able to recovery fast.
- We design a special architecture for CRNs, and propose the virtual backbone construction framework. The construction phase does not assume the existence of common control channel, which is more practical. Also, the corresponding MAC and routing protocols under the virtual backbone structure are studied.

The organization of the rest chapters of this dissertation is: we first review the relevant research works in Chapter 2. The schemes of improving the spectrum sensing performance are discussed in Chapter 3. Chapter 4 presents the channel assignment algorithms. The routing schemes are proposed in Chapter 5. We describe the architecture design for CRNs in Chapter 6. Chapter 7 concludes this dissertation and identifies the future research topics, which are potentially promising and challenging.

CHAPTER 2

LITERATURE REVIEW

This chapter provides the literature review regarding the works that are closely related to the research of this dissertation. We first start with the cooperation during spectrum sensing phase in CRNs. Then, we study the channel assignment approaches. We will also discuss the main routing protocols and route maintenance scheme for CRNs. In addition, we discuss about the current architecture design in CRNs.

2.1 Cooperation For Spectrum Sensing

Regarding the cooperation during spectrum sensing phase, there are two main types. One is about the cooperative spectrum sensing, which focus on the cooperation for the sensing techniques and improve the accuracy of the sensing results. The second one is about the recommendation based systems for spectrum sensing.

Cooperative sensing is studied in [6–8]. The main concept of cooperative sensing is that different SUs share their sensing results to achieve a more accurate result about the spectrum availability. Authors in [6] consider, in mobile CRNs, how to solve the problem of uncorrelated users to improve the cooperative spectrum sensing performance. [7] considers a soft combination of the observed energies based on the Neyman-Pearson criterion. An optimization scheme is proposed in [8], which uses a coordinator to make decisions. The work in [9] studies the optimization problem in cooperative spectrum sensing based on the sensing throughput tradeoff. The above works focus on the spectrum sensing phase, which is based on the collaboration among different SUs. Their key point lies in the fusion of different sensing results and the

throughput optimization.

A recommendation-based model is proposed in [10, 11]. The authors are inspired by the recommendation system for e-commerce, and they have each node recommend channels to other nodes for spectrum sensing. However, their model only considers the appearance of PUs in the recommended channels. In fact, channel availabilities depend not only on PUs, but also on SUs.

2.2 Channel Assignment

Optimal conflict-free channel assignment that satisfies a global optimal objective is often NP-hard [12]. Based on a simplified interference model, this problem can be described as a vertex-coloring or edge-coloring problem. The generalized form of our problem could be reduced to a list-edge-coloring problem [13], which assigns every edge to a color from a prescribed list. Centralized approximations in CRNs, such as [14, 15], formulate the problem as a mixed integer programming problem. Various distributed approximations are proposed. In the graph color model, distributed list-edge coloring algorithms usually rely on a modified version of edge-coloring algorithms. Several distributed algorithms have been proposed in literature [16–18]. In [19], Wang and Liu considered an iterative distributed solution, based on the orientation of each link. An end node with a larger number of channels points to another one with a smaller number of available channels. The channel assignment starts with nodes that are local minimum (i.e. the minimum number of channel choices) and applies this process iteratively. An end node with a larger number of channels points to another one with a smaller number of available channels.

2.3 Routing Protocols and Route Maintenance

The routing issues in CRNs mainly contain three aspects. The first aspect is about how to select routes in CRNs. Since the route selection is tightly closed to the channel assignment, the second one is about the joint routing protocols and channel assignment. The third one is how to perform route maintenance, e.g., route recovery in CRNs when the initial routes become broken due to PU activities.

2.3.1 Route Selections

The general approach to route selection consists of two phases: graph abstraction and route calculation:

- Graph abstraction phase refers to the generation of a logical graph representing the physical network topology. The outcome of this phase is the graph structure $G = (N, V, f(V))$, where N is the number of nodes, V is the number of edges, and $f(V)$ the function which allows to assign a weight to each edge of the graph.
- Route calculation generally deals with defining/designing a path in the graph connecting source-destination pairs. Classical approaches to route calculation widely used in wired/wireless network scenarios often resort to mathematical programming tools to model and design flows along multi-hop networks.

In CRNs, the weight of each link needs to be defined. Also, the availability of each link is not static. In [20], the authors focus on the case where the metrics for the horizontal links are proportional to traffic load and interference. A similar approach based on graph structures is proposed in [21], where a colored graph is used to represent the network topology. The colored graph $G_c = (N_c, V_c)$, where N_c is the vertex set (one vertex for each network device), and V_c is the edge set. Two vertices in the colored graph may be connected by a number of edges up to M , where M is the number of channels (colors) available for transmission on the specific link. The

route calculation algorithm follows the same rationale as the one proposed in [20], leveraging a centralized iterative approach. Once a flow has been routed, the colored graph is updated by re-setting the edge weights, then iterating for all the remaining traffic flows.

2.3.2 Joint Routing Protocols

In [22], the authors propose a protocol called opportunistic cognitive routing (OCR), which enables each user to make use of its geographical information and collect channel usage statistics. Specifically, each SU independently selects the next hop relay based on the local channel usage statistics so that the relay can quickly adapt to the link variations. In [23], the authors study a routing protocol called CRP, which maps the spectrum selection metrics and local PU interference observations to a packet forwarding delay over the control channel. [24] develops a centralized channel assignment scheme and bandwidth allocation combined with routing algorithms for multi-channel wireless mesh networks. [25] presents a routing and channel assignment protocol for multi-channel multi-hop wireless networks. It balances channels by having each node select channels based on its load information. In [26], the authors consider a distributed channel assignment and routing scheme in multi-channel multi-hop wireless networks. The authors in [27] focus on the joint routing and link scheduling problem for multi-hop CRNs under uncertain spectrum supply. In addition, there have also been some works done to take the angle dimension of CRNs into consideration[28–30].

2.3.3 Route Maintenance

The route maintenance happens when some links are broken during the sudden appearance of PUs. There are usually two choices, handoff and rerouting, for each node to fix the broken links. The handoff is to switch to another channel, and re-

coordinate with the receiver. The rerouting is to perform routing again and find another route to reach the next hop node. Different approaches are proposed based on different route maintenance goals.

In [31], Feng et al. propose a handoff scheduling and rerouting scheme. Their protocol assumes the predictability of the PUs and has each flow ready to reroute before the PUs appear. The delays are not considered. In [32], Abbagnale et al. focus on providing a route with more stability. The main idea is to assign weights to routes based the algebraic connectivity. A metric is proposed that can capture path stability and availability over time. Their approach is based on the historical average activity model of PUs. The concept of route maintenance cost is proposed in [33]. The cost represents the effort to maintain the end-to-end connectivity in CRNs. Their approach starts to obtain an optimal path with minimum route maintenance cost under the perfect knowledge of PUs. They define the maintenance cost by representing the effort needed or penalty paid to maintaining end-to-end connectivity. Works in [34–36] consider the routing delay, which contains not only the data transmission delay, but also the route maintenance delay. The switching delay is proposed in [34, 35], which considers the switching delay between different channels and the backoff delay (medium access delay) with a given channel. Each node maintains a metric of the cumulative delay along a candidate route. In [36], Yang et al. also propose a distributed method for local coordination. Their on-demand protocol is a variation of AODV.

2.3.4 Architecture Design in CRNs

The related researches in this part are divided into two parts. One is about the cluster applications in CRNs. The other one is about how to exchange messages for architecture construction without common control channels in CRNs.

Many recent works have been done on the cluster structures in CRNs [37, 38].

In [37], the cluster structure is used on the allocation of control channels in CRNs. Different channels for control are allocated at various clusters in the network, and the problem is solved by the bipartite graph. Authors in [38] construct clusters according to the event that happened in cognitive radio sensor networks. The detection of an event forces the eligible nodes in the network to form into clusters, and better deliver the message. The above models apply cluster structures in CRNs, and solves the control channel or data transmission related problems.

The main challenge for architecture design lies on the channel rendezvous phase, which is for control message exchanges. Several approaches to building links in CRNs without CCC were proposed. [39] proposed an algorithm with the nodes rendezvous time as a function of the number of channels, while the algorithm in [40] has rendezvous time as a function of the number of nodes. Both are based on the assumption of some pre-known network information, such as the number of nodes. Our model does not rely on the assumptions of node numbers, and provides an efficient learning process for nodes to know about each other. Our work focuses on building virtual backbones for end-to-end data transmission use. The authors in [41] proposed a rendezvous algorithm based on the quorum systems. However, in this approach, many nodes need to compete for one rendezvous channel. A jump-and-stay model was proposed in [42]. It is a blind channel rendezvous model. The approach in [43] utilizes the channel diversity and allows all channels to be a control channel. The work in [44] proposes the quorum and Latin squares channel Hopping scheme, which has each node to guess the sequence of the other nodes based on node IDs.

CHAPTER 3

SPECTRUM SENSING IMPROVEMENT IN CRNS

Since the sensing phase is inevitable for CRNs, how to improve the sensing performance while reducing the sensing time cost becomes a very important issue. In this chapter, we focus on the pre-phase of the spectrum sensing and discuss about the sense-in-order model. Then, we propose two structures, cores and clusters, to assist the spectrum sensing.

3.1 Pre-phase for Spectrum Sensing

We first introduce the state transition diagram for every channel on each node. Then, we describe how to determine the sensing order among different channels for each node, based on their states. In addition, we propose an improved model with angle dimensions taken into account. Finally, the performance evaluations are studied.

3.1.1 Preliminaries

Since there are two types of users, which are PUs and SUs in CRNs, the channel availabilities for a single node are determined by both types of users. First, the PUs are the privileged users, which have higher priorities to use the channels. Obviously, the channels occupied by PUs are unavailable. Regarding the channels that are not occupied by PUs, they are not available for sure. This is because other SUs may

be transmitting on those channels. A channel is only available when it is occupied neither by PUs nor SUs.

The difference of a channel to be occupied by a PU or a SU lies in whether other nodes can be notified when the user finishes using that channel. For a channel being occupied a SU, since SUs can communicate over the common control channel (CCC), when the SU finishes its transmission and quits that channel, it is able to broadcast this information to other nodes. However, it is impractical for PUs to communicate with SUs and the active duration of PUs are unpredictable. Therefore, for channels being occupied by PUs, it is more difficult for SUs to know when those channels become available again.

When a node in a CRN tries to predict availabilities of all channels based on its collected information before spectrum sensing, it needs to combine the information from different dimensions. First, the frequency dimension needs to be considered since there are multiple channels in CRNs and each channel must be considered separately. Second, the time dimension is also important. Since the node needs to predict channel availabilities based on its previously collected information, the time of when the channel information is received needs to be considered. This is because the channel availabilities are dynamic in CRNs. A channel may be unavailable at the next time slot even though it is available in the previous time slot. Therefore, the channel information received more recently is more reliable.

In our extended model, we also consider one more dimension, the angle dimension. With the help of directional antenna technology, each node is able to transmit only on some certain direction. The directional antenna brings more channel opportunities in CRNs. In addition, it introduces more complexities since each node needs to predict the channel availabilities and choose a channel for sensing in each direction.

3.1.2 Construct State Transition Diagram

Spectrum sensing in CRNs is an important phase, in which SUs determine if a channel is available. Lots of work has been done on how to achieve an accurate result through spectrum sensing[45–49]. There are usually multiple channels in CRNs. However, each SU cannot sense all of the channels simultaneously. Therefore, each time a SU needs to find one channel for transmission, it will pick one channel for sensing. If the channel is unavailable, it needs to adjust its parameters and switch, as to sense another channel. For example, in Fig. 3.1, there is a pair of PUs. One is the transmitter, denoted as TX and the other one is the receiver, denoted as RX . The SU u is in the interference range (the amoeba shape) of TX , which is using the channel m_1 to send data to RX . There is a total of three channels (m_1, m_2, m_3) in the network. If u needs to use one channel, it will pick one channel from the three channels. Since there are no differences among the three channels from u 's point of view, it is possible that u will pick m_1 for sensing. Then, after u finds out that m_1 is unavailable based on the sensing results, it needs to switch to another channel and sense again. However, if u can have some information about the three channels before sensing, it may avoid m_1 and select other channels for sensing. To some extent, both the delay and the energy consumption can be reduced.

Our focus here is not the spectrum sensing technology itself. Instead, we consider how to choose a channel for sensing for each node at the beginning, so that the probability of switching to sense another channel is reduced. This is also called the pre-phase of spectrum sensing. We propose a sense-in-order (SIO) model, which provides each node with an order for spectrum sensing. The order is determined before the spectrum sensing, and is maintained in the form of a list by each node. When a node needs to find a channel for data transmission, it can look up the list and select a channel that has a higher probability of being available for sensing. In this way, each node knows the order in which to sense; this results in a reduction of

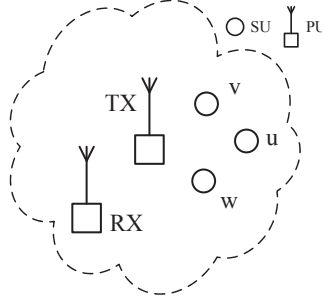


Figure 3.1: Sense-in-order can reduce delay and save energy.

switches among channels during spectrum sensing.

To determine the order of channels for sensing, both the space and time dimensions are considered. Firstly, the space factor influence lies in that nodes in a similar geographical area usually share similar channel information. One node can broadcast its sensing results to other nodes. Secondly, the time dimension is also very important, due to the dynamics of PUs. For each node, the channel information received more recently is more likely to be accurate. Based on the information from both dimensions, we identify four different states and their transitions for every channel, which are maintained on each node. Based on the states of different channels, each node is able to divide the whole channel into several different subsets. Each subset is assigned with a probability of being chosen. We also take conflict avoidance into account, since nodes in close locations are very likely to choose the same channel for sensing.

3.1.3 Sense-in-order Model

Each node broadcasts its sensing results through the CCC and does not sense the channel until it needs a channel for transmission. Thus, if the node finds an available channel, it will access that channel. It will also broadcast when it accesses and when it quits that channel. This can be done by broadcasting different signals. We give the following three different types of signals that can be sent by a node u :

- PO_m : channel m is occupied by PUs;

- SO_m : channel m is free from PUs, but is occupied by the SU who sent this signal;
- SF_m : SU finishes transmission and quit from channel m .

To simplify our mode, we assume that there is no loss of the signal transmission over the CCC. If channel m is occupied by SUs, node u will avoid sensing that channel since SO_m was received previously by u until the SF_m is received. Based on the received signals, a node v is able to identify four different states, $S = \{S_i, 1 \leq i \leq 4\}$, for a channel m . We use $\langle S_i, m \rangle$ to indicate that channel m is in state S_i :

- $\langle S_1, m \rangle$: m is occupied by PUs;
- $\langle S_2, m \rangle$: m is not occupied by PUs, but is occupied by the SU;
- $\langle S_3, m \rangle$: the SU previously using m has finished transmission and quit from m ;
- $\langle S_4, m \rangle$: no signal is received about m .

The above four states are maintained on node v itself. For $\langle S_1, m \rangle$, node v is not sure about whether the PUs have finished transmission on m if no other sensing results are received from other nodes. For $\langle S_2, m \rangle$, node v should avoid sensing m until v receives the signal SF_m . For $\langle S_3, m \rangle$, node v should assign higher probabilities for selecting m to sense. For $\langle S_4, m \rangle$, v is not sure about the availability of m either.

Furthermore, the weight of each signal about its accuracy should consider the following two issues:

- spacial domain: The channel state information that is received from the closer area is more accurate. Therefore, the channel state information that is sensed by the node itself is usually more reliable than the information shared by others.

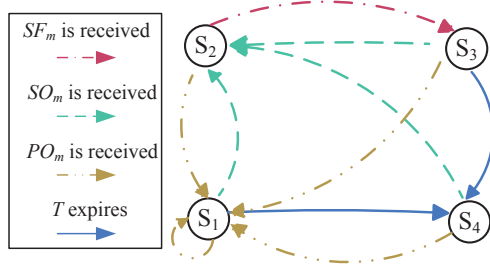


Figure 3.2: State transition diagram.

- time domain: The message that is received more recently is usually more accurate, which is due to the dynamics of PUs' and SUs' activities. Some channels that were available in the previous time slot may possibly become unavailable in the next time slot.

In our model, each node only collects one-hop neighbors' information and updates the channel states. Since the interference area of a PU is usually much larger than that of a SU, nodes within one-hop distance may very possibly share similar information. This is normally true in real life scenarios. Therefore, there is no need to distinguish the weight of the information shared by a node's neighbors based on their distance to this node. In our simulation, we will study the influence of spacial factors.

The time domain matters more than the spacial domain. Firstly, as we explained above, the information that is received more recently is more reliable. The channel state should be updated according to the most recently received information about that channel. Secondly, the channel states may vary without being known by any node. Therefore, we add a valid time period T for S_1 , S_2 , and S_3 . If no signal is received about channel m during T , node v will change from $\langle S_1, m \rangle$, $\langle S_2, m \rangle$, or $\langle S_3, m \rangle$ to $\langle S_4, m \rangle$.

The state transition diagram among different S_i of a single channel m maintained on node v is shown in Fig. 3.2. The initial state of each channel is S_4 . A state transition occurs when a signal from another node is received, or the valid time period T expires. Each node maintains a state identification for each channel, and updates

the state based on Fig. 3.2. Each state is updated based on the most recent signal in the time domain. Since PUs have higher privileges on each channel, PO_m could be received no matter what the previous state is. State S_3 can only be reached from S_2 , since we assume that there is no packet loss. Each node will mark the channel as S_2 after receiving SO_m , and will not update the state as S_3 or S_4 before SF_m is received. Thus, there is no valid time expiration issue for state S_2 .

3.1.4 Sort Sensing Order

Since each node stores the state for every channel, it now can define preferences on different channels when it needs to select one channel for sensing.

First, each node divides the whole channel set into four (at most) different subsets, based on the state of each channel. For node v , the whole channel set M is divided into four subsets $M_v(S_i)$, $1 \leq i \leq 4$. If channel $m \in M_v(S_i)$, channel m is identified as state S_i by node v . Next, we define the order or probability for each subset to be chosen. Here, the probability of a subset being chosen equals the sum of the probability that any channel in that subset will be chosen.

Definition 3.1. The probability of $M_v(S_i)$ being chosen is:

$$P_v(S_i) = \frac{|M_v(S_i)| \times W_i}{|M|},$$

where:

$$i = \{1, 2, 3, 4\}, \tag{3.1-1}$$

$$W_2 = 0, \tag{3.1-2}$$

$$W_3 > W_4 > W_1, \tag{3.1-3}$$

$$\sum_i (|M_v(S_i)| W_i) = |M|. \tag{3.1-4}$$

Algorithm 1 Order of channels for sensing

- 1: **while** v is in the network **do**
 - 2: **if** a signal about m is received over CCC or T_m expires **then**
 - 3: Update the state for m based on Fig. 3.2
 - 4: **if** v needs to transmit **then**
 - 5: pick the set $M_v(S_i)$ based on $P_v(S_i)$
 - 6: pick a channel $m \in M_v(S_i)$ based on p_v^m
-

$|M_v(S_i)|$ denotes the number of channels in $M_v(S_i)$. (2) means that the channels identified as state S_2 will not be chosen for sensing. This is because these channels are definitely unavailable, according to the discussions in previous sections. The relationships in (3) are due to the fact that channels on state S_3 are more likely to be available than channels on S_1 and S_4 . W_i is the weight assigned to choosing each channel set $M_v(S_i)$. This means that, a channel is sensed to be occupied by PUs, it is still possible that it is available when sensed by node v , since PUs at that time may finish a transmission. (4) ensures that $\sum_i P_v(S_i) = 1$.

For different channels in $M_v(S_4)$, the probability of each one being chosen is the same. For different channels in $M_v(S_1)$ and $M_v(S_3)$, the probability of a single channel m being chosen should also be related to the amount of time that m has been in that set. The more time that m is in $M_v(S_1)$ or $M_v(S_3)$, the less accurate that the state of m will be. Node v maintains a time duration, t_m , to indicate how long m has been in that state. Then we define the probability that a single channel m will be chosen:

$$p_v^m = \begin{cases} \frac{t_m}{\sum_{m_0 \in M_v(S_1)} t_{m_0}} \times P_v(S_1) & m \in M_v(S_1) \\ 0 & m \in M_v(S_2) \\ \frac{T-t_m}{\sum_{m_0 \in M_v(S_3)} (T-t_{m_0})} \times P_v(S_3) & m \in M_v(S_3) \\ \frac{P_v(S_4)}{|M_v(S_4)|} & m \in M_v(S_4) \end{cases}.$$

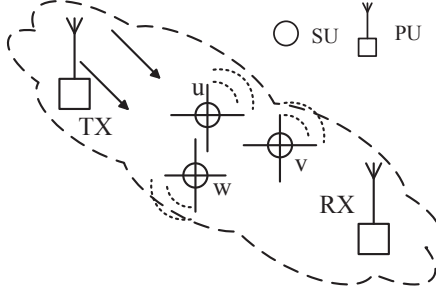


Figure 3.3: Directional antenna creates more channel availabilities.

For a channel $m \in M_v(S_1)$, the longer t_m means that m is more likely to be available, since PUs have a higher probability of finishing with m . For a channel $m \in M_v(S_3)$, the less t_m indicates that m has a higher probability of being available. When t_m grows larger, there is an increased possibility that m may be taken by PUs without notifying node v .

Algorithm 1 is for a node v to define the order of channels for sensing. Node v keeps overhearing the CCC and monitoring if T_m expires for each m . When v needs to select a channel for sensing, it first decides which subset $M_v(S_i)$ to choose. Then, from the picked $M_v(S_i)$, it chooses the channel for sensing. This maintains the priorities among different $M_v(S_i)$, which are stated above. Also, since $\sum_{m \in M} p_v^m = 1$, there is always one channel that can be chosen when a node senses the channel.

Since nodes in a close area maintain similar states for each channel, when multiple nodes among them need to pick a channel for sensing at the same time, it is very possible for them to choose the same channel. However, in our model, the chance of conflicts can be reduced by adjusting W_i in the expression of $P_v(S_i)$. If conflicts occur more often when choosing channels from $M_v(S_i)$, the W_i is reduced. We will show the effects of adjusting W_i on the whole cost in our simulation.

3.1.5 Extended Model for SIO

We consider a model in which each SU is equipped with a directional antenna that is able to work in four different directions. We assume that each node can send

over one direction, while overhearing from four directions through the CCC on one channel. As shown in Fig. 3.3, there are two PUs, TX and RX, who are active and occupy a channel. The SUs, u , v , and w , are located in the interference area of the PUs. Instead of being unable to use the channel that is occupied by the two PUs, with the directional antennas, they are now able to use that channel from one of the four directions. Therefore, the channel availabilities are improved for each SU.

With increased channel opportunities, the introduction of the angle dimension is problematic to our sense-in-order model. Next, we focus on how to determine whether one channel is available in one direction, based on the information provided by neighbors. For any state S_i , $S_i \in S$, we use 1 to denote that S_i is true and 0 to denote that S_i is false on each direction for every channel. The above state transition diagram can be easily extended here through adding two substates for each S_i and corresponding signals on each direction. Here, we concentrate on how to determine if a single state S_i is true or false for each direction.

For a single node, if it receives a signal of a state in one channel, the node will set the state of that channel to 1. Otherwise, it will set it to 0. We use $aaaa$ to denote whether a state is true in four quadrants. Each quadrant denotes a direction. Obviously, $a = 0$ or 1 . The problem is for a single state, it is possible that one node can have 16 different sequences from four directions regarding the same channel. For example, 1110, 0011, etc. This is because each direction is adjacent to another while the interference area may cover part of multiple directions together. Nodes at different positions of each direction may broadcast different signals simultaneously. When one node hears different sequences within a close time period, it may be difficult to decide which direction is available.

The method we use here is a weight-based scheme, which considers both the number of signals received and the time at which the signal was received. As we explained before, the signal that is received more recently is more accurate. We use

Table 3.1: Simulation settings for SIO evaluation.

number of nodes	[10, 18]
number of channels	[4, 12]
number of PUs	[8, 14]
session duration of SUs	[5, 10]
session duration of PUs	[10, 16]
valid time period, T	20
T_{SC} for SIO-SC	[0, 15]
W_3/W_4	[2, 3]
W_4/W_1	[1.5, 2.5]

TT to denote the time period since receiving the signal 1, and TF to denote the period since receiving the signal 0, on each direction regarding each state. We also define a valid time period, T' , for each signal. Signals received before T' will be ignored. Then, each node is able to calculate the probability of a state to be true or false on each direction for one channel by using the following equation:

$$PT = \frac{\sum_{\forall TT < T'} (T' - TT)}{\sum_{\forall TT < T'} (T' - TT) + \sum_{\forall TF < T'} (T' - TF)};$$

$$PF = \frac{\sum_{\forall TF < T'} (T' - TF)}{\sum_{\forall TT < T'} (T' - TT) + \sum_{\forall TF < T'} (T' - TF)}.$$

Since $PT + PF = 1$, each node determines that a state is true on one direction for one channel if $PT > 0.5$, and false if $PF \geq 0.5$. After the state is decided, the previous SIO model can be applied for each direction.

3.1.6 Simulations

We randomly generate a number of nodes which make up a one-hop network. Each node gets a randomly received session request, and stays in that session for several time slots. For a node, when a session request comes, it needs to pick a channel to sense. Also, we generate a number of PUs who randomly become active. The simulation setting parameters are shown in Table 3.1.

Considering that the objective of our model is to minimize the number of switches among channels during spectrum sensing according to Definition 1, we measure the performance of our model using the success percentage, which is defined as the ratio between the number of times that available channels are sensed and the total number of times that we attempt to sense channels.

We also implement two other algorithms:

- The basic algorithm: SIO-SO (sense-in-order model with self information considered only). Nodes running SIO-SO do not share their sensed channel information. Each node only uses its sensed channel information and history to decide each channel's state.
- The self weighted more algorithm: SIO-SC (sense-in-order model with self information weighted more). The SIO-SC takes the spacial factor into consideration. Each node running SIO-SC assigns more weight to the channel information sensed by itself, since it is one that is "closest" to itself. We assign a valid time window, denoted as T_{SC} , where $T_{SC} < T$. Any signal received from its neighbors within T_{SC} regarding the same channel state information would be ignored. If any signal is received between T_{SC} and T , the channel state would be updated according to the received signal. SIO-SC is the same as SIO without antenna when $T_{SC} = 0$.

We first compare the four algorithms, SIO with Antenna, SIO without Antenna, SIO-SC and SIO-SO. We vary the three network parameters (number of nodes, number of channels, and number of PUs) and calculate the average success percentage in the whole network. T_{SC} for SIO-SC is set as 10. The results are shown in Fig. 3.4. We can tell that the SIO with antenna model achieves the best performance while the SIO-SO is the worst one. Also, in Fig. 3.4(a), the performances of all four algorithms increase when the channel number increases. In Fig. 3.4(b), when the number of

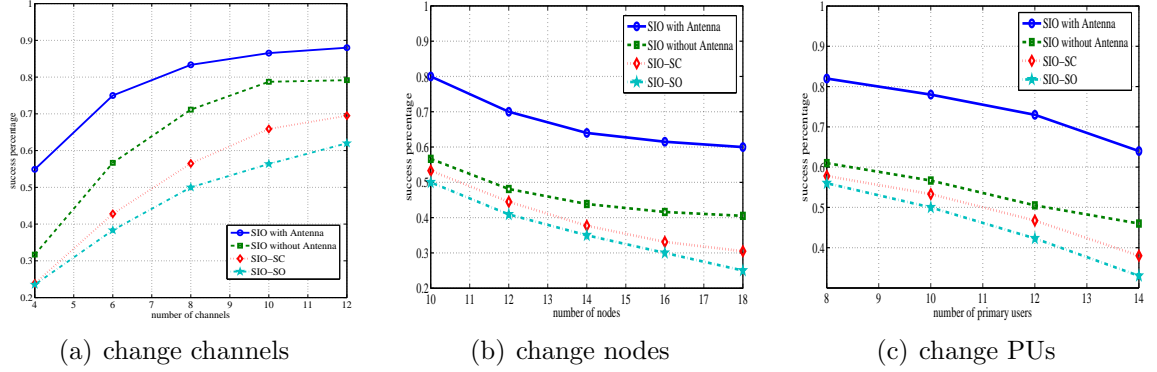


Figure 3.4: Comparison of success percentage when varying network settings.

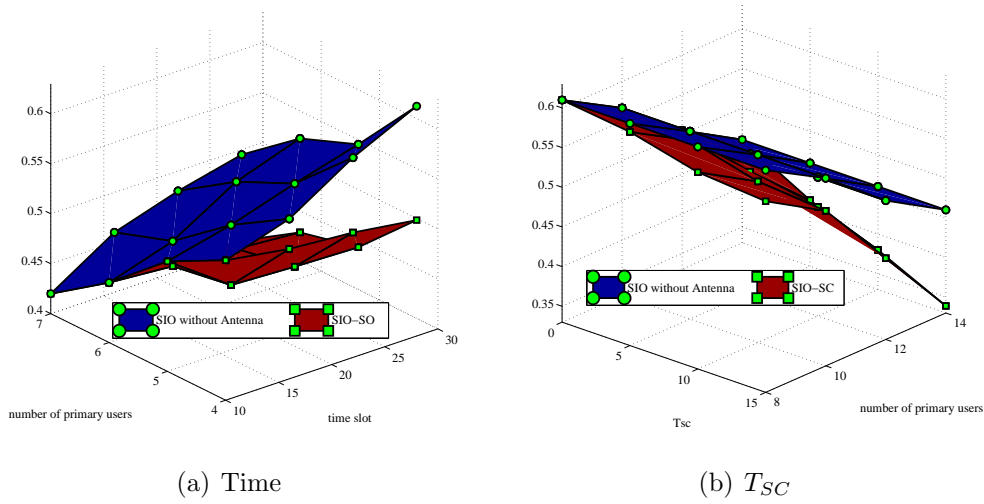


Figure 3.5: Study of parameter influences

nodes increases, the performance of all four algorithms is lowered. In Fig. 3.4(c), the performances of all four algorithms decrease when the amount of PUs increases.

For a better comparison, we also study the performance differences between the SIO without antenna model and the SIO-SO scheme over a time period. The SIO with antenna scheme has a similar trend as the one without antenna, which is not shown here. We vary the number of PUs and record the success percentage along a time duration. The results are in Fig. 3.5(a). It shows that when the PUs decrease, the performances of both schemes increase. However, the performance of the SIO-SO scheme increases more slowly than our model. The influence of T_{SC} on SIO-SC is

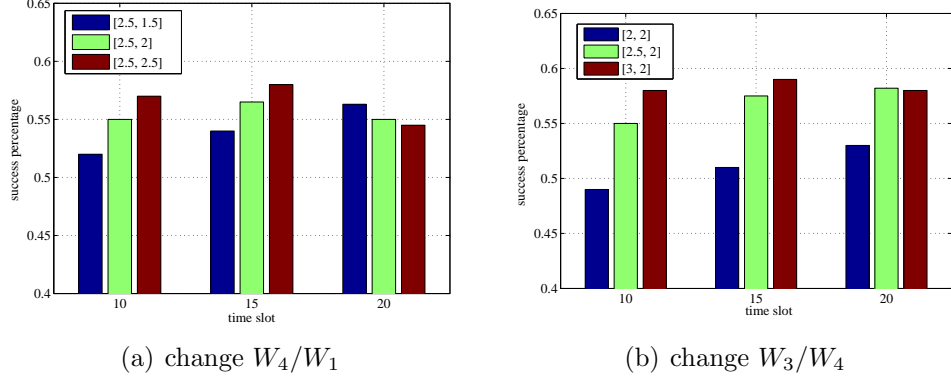


Figure 3.6: Comparison of success percentage when varying algorithm settings.

studied in Fig. 3.5(b). The performances of both algorithms decrease as the PUs increase. When the value of T_{SC} increases, the gap between the two algorithms increases.

The influences of the algorithm parameters are shown in Fig. 3.6. The SIO with antenna is not shown here due to the same reason as above. We calculate the success percentage at different time slots when running the simulation. In Fig. 3.6(a), the value of W_3/W_4 is set as a constant 2.5, while the value of W_4/W_1 varies in $\{1.5, 2, 2.5\}$. It shows that when the time increases, the performances of the three settings increase. In Fig. 3.6(b), the the value of W_3/W_4 varies in $\{2, 2.5, 3\}$, while the value of W_4/W_1 is set as a constant 2. The performances of the three settings are better when the time slot increases.

3.1.7 Conclusion

In this section, we focus on how to choose a channel for sensing for each node in CRNs. We propose an SIO model for the pre-phase of spectrum sensing. We construct a state transition diagram and a corresponding algorithm for each node to calculate the probability of each channel being chosen for sensing. We also extend our SIO model by adding the angle dimension. The simulation results demonstrate that our model outperforms the others.

3.2 Hitchhiking for Sensing

In this section, we make use of the fact that nodes located in close geographical locations share similar channel information or sensing results at a given time. Our model enables each node to hitchhike in its spectrum sensing phase, by using other nearby nodes' recent sensing results. We propose two frameworks for assisting nodes in selecting channels for spectrum sensing. One is the core-only structure, and another is the mixed cluster-core structure. We present the construction of two structures and the distributed spectrum sensing schemes, respectively.

3.2.1 System Model

We consider a CRN with multiple PUs, which are randomly distributed. The total channel set in the network is M . There are V nodes (SUs), which opportunistically make use of the channels without interfering with PUs. The PUs are randomly distributed in the network. Their locations and active patterns are assumed to be unpredictable. Each PU has its own privileged channel in M . When a PU becomes active, it would occupy its channel. First, we assume that there is a CCC for nodes in CRNs to exchange information.

Every time slot is divided into two parts, the spectrum sensing and data transmission. For a node v ($v \in V$), before transmitting data on a channel m_v ($m_v \in M$), it needs to perform spectrum sensing on m_v , to make sure PUs of m_v are not interfered with. We say that m_v is available on node v , if v 's transmission on m_v does not interfere with any PU on m_v . For channels that are free of PUs but occupied by other nodes, v can compete for access equally with other nodes. A channel that is free of both PUs and other nodes is the most ideal, since no or less competition is needed. The goal of the spectrum sensing by v is to find an available channel. It is better if the channel is also free of other nodes, but this is not required. If all the channels are occupied by PUs, v cannot transmit data until one channel becomes available.

Unlike some spectrum sensing models, our model does not have a centralized decision fusion center. This is because, the PUs are randomly distributed in our model with different transmission ranges. It means that the channel availabilities are not the same in the whole network. Therefore, a centralized decision fusion center is unnecessary.

We assume that the average cost for sensing one single channel is a constant γ . Then, the spectrum sensing performance is measured by the number of channels that a node v needs to sense, denoted as X_v , until one available channel is found. The objective of our model is to minimize the spectrum sensing cost, which can be formulated as:

$$\text{Min}(\gamma \cdot X_v), \forall v \in V. \quad (3.2-5)$$

For a channel to be successfully used for data transmission, it should be free of PUs. Moreover, the node does not want to use a channel that is also chosen by many other nodes, since there would be too much cost spent in the competition for channel access. Therefore, the goal of each node sensing the channels is to find one that must be free from PUs, and better to be free of other nodes may cause interferences to it. Our focus here is to reduce the number of channels that need to sense for each node to find a channel for data transmission, instead of the sensing technique itself. Therefore, we assume that the sensing is precise enough for each node to detect PUs, and to decide if one channel is occupied by PUs or other nodes. We will discuss our model to reduce X_v in Eq. 3.2-5, since γ is a constant here. There is no optimal solution because of the fact that the PUs' activities are dynamic and cannot be pre-known. We give our heuristic solutions, while bringing limited extra cost.

3.2.2 Core-Only Structure

Our distributed core construction approach is designation-based and only requires one-hop information. The process overview is described below:

1. Each node v exchanges its current available channel set M_v with nodes in its neighbor set N_v ;
2. Based on the gathered information in Step 1), node v calculates its weight, which is related to the size of both its available channel set M_v and neighbor set N_v ;
3. Node v exchange its weight information with nodes in its neighbor set N_v , and designate the one neighbor with the highest weight (can be v itself), to be its core.

To present the above process in detail, first, we give the definition of weight for each node, which acts as a basis for later core selection.

Definition 3.2. For a node v , the weight of it is defined as $W_v = \sum_u |M_u \cap M_v|$, $\forall u \in N_v$.

$M_u \cap M_v$ is the intersection of nodes u and v 's available channel sets. Based on the above definition, the nodes with more neighbors, and more common available channels with their neighbors, have a larger weight. Having the weight defined, the core definition is:

Definition 3.3. c_v is the core of node v , if and only if $c_v \in N_v$ and $W_{c_v} \geq W_u$, $\forall u \in N_v \cup \{v\}$.

After each node exchanges its weight with all its neighbors, then every node can designate the core node from its own perspective. Of course, for a node v , it can have its own core, and can also be selected by some of its neighbors as their core. The set of those nodes that select v as their core are denoted as D_v . In other words, if $u \in D_v$, then $c_u = v$. We say that D_v contains members of core v .

The algorithm for a node v to get c_v and D_v , is given in Algorithm 2. The exchanged information, including the core selecting decision, is sent through the CCC.

Algorithm 2 Find c_v and D_v for node v .

Require: $N_v, \{W_u, \forall u \in N_v \cup \{v\}\}$;

Ensure: c_v , core of v ; D_v , set of nodes selecting v as core;

- 1: $Marked_v = false, D_v = null, c_v = null$;
 - 2: Find $max(W_u), \forall u \in N_v \cup \{v\}$;
 - 3: $c_v =$ the node has $max(W_u)$;
 - 4: v sends its selecting decision to c_v ;
 - 5: $Marked_v = true$;
 - 6: **for** every $u \in N_v \cup \{v\}$ **do**
 - 7: **if** $Marked_u = false$ **then**
 - 8: Wait until $Marked_u = true$;
 - 9: **if** $c_u = v$ **then**
 - 10: $D_v = D_v \cup \{u\}$;
 - 11: **return** c_v, D_v ;
-

There is no order constraint, which means nodes can run Algorithm 2 in parallel. For node u , c_u is the core of u , which has the maximum weight among all nodes in N_v . It also scans its neighbors, and put the nodes that select v as their core in D_v . Node v ends the algorithm until all the nodes in N_v have selected their cores. If node v does not have any neighbor, it would designate itself, which means $D_v = \{v\}$ and $c_v = v$. Every node is covered by a core structure after running Algorithm 2.

An example of the above core construction is shown in Fig. 3.7. There are 7 nodes and neighbor nodes are connected. The total channel set M is $\{1, 2, 3\}$. The table in Fig. 3.7 lists the available channel set of each node on the second row and its corresponding weight on the third row, calculated according to Definition 3.2. There are two cores here, which are v and u , marked in black. The node set that designates v and u as the cores are shown as d_u and d_v in Fig. 3.7.

From the above core construction, it may result in too many cores in the network, and each core only has a very small number of members. This could reduce the efficiency of our core structure when assisting the spectrum sensing, as introduced in the next part. Moreover, the number of members for each core cannot be too large. Otherwise, the overhead of the communication among each core and its members would be too much. Therefore, for a core v , we add the size constraints S_{min} and

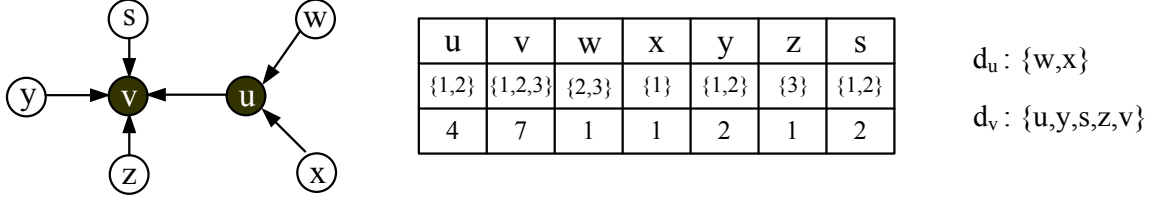


Figure 3.7: An example of constructing core structure.

S_{max} , where S_{min} and S_{max} are minimum and maximum size of D_v . For node v , if the core c_v chosen by Algorithm 2 does not fit the size constraint, the selected c_v would be removed from the original neighbor set $N_v \cup \{v\}$. A new core would be selected from the rest of the nodes in N_u , which satisfies the size constraints. If no nodes in N_v are left, or v itself is chosen as its own core.

3.2.3 Spectrum Sensing with Cores

After the core structure of the network is constructed, every node will gain help from its core for spectrum sensing. This basic idea is that, when a node has data to transmit, it will ask its core about which channel to sense so that the total spectrum sensing cost is reduced. The core node needs to maintain such information to give assistance when help requests are received. We will discuss our scheme from the node side and the core side.

On the node side: When a node has data to transmit, the works that it needs to do are divided into four categories, which contain the four actions:

1. *Pull*: pull the latest channel list from the core, which sorts channels according to their recent updated information;
2. *Sense*: sense the channels using the given order in the list, until one available channel is found;
3. *Transmit*: perform data transmission on the selected channel, and repeat *sense*

again if the current channel becomes unavailable because of PUs;

4. *Push*: After data transmission is completed, push the updated channel situations to the core, along with the updated time. The information contains all the channels the node sensed during the above phases.

If there are multiple nodes sending requests to their common core at the same time, the interfering nodes would back off, and send again until receiving the list from the core. When a node has a certain amount of data to transmit, the *pull* and *push* only happens at the beginning and the end of the data transmission. The frequency of exchanging information between a node and its core is changeable.

It is possible that a node can also be a core. For a node v , if $c_v = v$, which means v is its own core, then v can simply sense based on the channel list maintained by itself. If v is a core to other nodes, but $c_v \neq v$, then v provides the channel list for nodes that pick v as the core, but v itself still needs to interact with its own core. For example, in Fig. 3.7, $c_w = u$ and $c_u = v$. The node w interacts with u to get and update the channel list, while u interacts with v . The two channel lists are independent.

On the core side: When the core receives a help request from its members, it needs to retrieve the current channel list and return it to the request sender. Also, when the core receives channel updates from its members, it needs to update the channel list for use next time.

Suppose $c_u = v$, the interaction between them is shown in Fig. 3.8. Having clarified the process between a node and its core, the remaining problem is this: How is the channel list maintained and updated by a core? To begin, we need to find a way to calculate the priority of each channel for the core to decide the channel order in the list. First, we associate each channel m , where $m \in M$, with 3-tuple attributes, $\langle TA(m), TN(m), TP(m) \rangle$:

- $TA(m)$: The last time when the core receives an update which reports m as

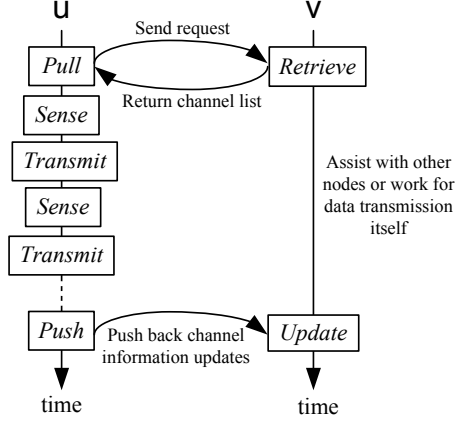


Figure 3.8: The interaction example between u and v ($c_u = v$).

free from PUs and other nodes;

- $TN(m)$: The last time when the core receives an update which reports m as occupied by other nodes;
- $TP(m)$: The last time when the core receives an update which reports m as occupied by a PU.

Based on the 3-tuple attribute of each channel, for a core v , it can sort the channels in M , which is shown in Algorithm 3. Every time v receives a help request from any node in D_v , it will run Algorithm 3 and return the sorted channel list, denoted as $L_v(M)$. The sorted results are three categories in $L_v(M)$. The first category contains the channels that are most recently updated as free from both PUs and other nodes. The second category contains the channels that are most recently updated as occupied by other nodes. Although these channels can also be used, the competition cost for channel access with other nodes could potentially be very large. The third category contains the channels that are most recently updated as occupied by PUs. Among channels in the same category, they are sorted according to their recent update times. The one that is most recently updated takes the more prior position in $L_v(M)$, since it is more reliable.

Algorithm 3 Calculate $L_v(M)$ by core v .

Require: $\langle TA(m), TN(m), TP(m) \rangle, \forall m \in M$;

Ensure: $L_v(M)$, the sorted channel list by their 3-tuple attributes.

```

1:  $p_c = 0$ ; {Point to the current index.}
2:  $p_t = 0$ ; {Point the tail indexes of previous part.}
3: for every  $m \in M$  do
4:    $\Delta_m = \text{Max} \{TA(m), TN(m), TP(m)\}$ ; {Get the last updated time.}
5: for  $i = 1 : 3$  do
6:   for every  $m \in M$  do
7:     if  $(i = 1 \ \& \ \Delta_m = TA(m)) \parallel (i = 2 \ \& \ \Delta_m = TN(m)) \parallel (i = 3 \ \& \ \Delta_m = TP(m))$  then
8:       Insert  $m$  to the position  $p_c$  of  $L_v(M)$ ;
9:        $p_c = p_c + 1$ ;
10:  Sort  $L_v(M)$  between positions  $[p_t, p_c]$  based on  $\Delta_m$  descendingly; {The more recently updated channel is in a more prior position.}
11:   $p_t = p_c$ ;
12: return  $L_v(M)$ ;

```

For example, in Fig. 3.7, if u sends a help request to v , v will run Algorithm 3 to sort the channels. Suppose the 3-tuples associated with the three channels are: channel 1, $\langle 13, 8, 9 \rangle$; channel 2, $\langle 11, 10, 7 \rangle$; channel 3, $\langle 5, 6, 12 \rangle$. Channels 1 and 2 are in the first category, and channel 3 is in the third category. Therefore, the sorted results of the three channels will be $L_v(M) = \{1, 2, 3\}$. Node v will return the $L_v(M)$ to u . After u gets the $L_v(M)$, it first senses channel 1 at time 16, and finds that it is occupied by a PU. Then u records this information and senses channel 2. If channel 2 is available, u will use channel 2 for data transmission. Suppose u finishes its data transmission at time 20, and the channel 2 is free of PUs during the data transmission, which means u only needs to keep sensing channel 2 in each *Sense* phase. Then u quits channel 2, and channel 2 is available at time 20. During the push phase, u will push the newly updated channel 1 and 2's information to v , along with the update times, 16 and 20. Node v then updates channel 1's 3-tuple as $\langle 13, 8, 16 \rangle$, and channel 2's 3-tuple as $\langle 20, 10, 7 \rangle$. Next time, when v receives another help request, the returned $L_v(M)$ is $\{2, 3, 1\}$.

3.2.4 Core Evolution

It is not always the case that one node shares similarities with other nodes which initially select the same core. For example, in Fig. 3.7, nodes $\{u, y, s, z, v\}$ designate v as their core. When u sends a help request to v , v needs to return the $L_v(M)$ to u . Assume that nodes $\{y, s, z, v\}$ have sensed channels before u . Then, based on Algorithm 3, the $L_v(M)$ is updated according to the sensing results provided by $\{y, s, z, v\}$. However, it is possible that the channel available situation around u , and nodes in $\{y, s, z, v\}$, are not very similar, due to the factors on interference range boundaries of PUs and the different geographical terrains. Therefore, it is inappropriate and inefficient for u to designate v as its core, which means it is necessary for u to find a new core that can help more on its spectrum sensing.

How does a node know if it designates a wrong core, itself? The answer is to evaluate the help that the core can provide. If the channel list provided by the core does not increase the spectrum sensing performance, the node should designate a new core. We use the estimated number of channels that a node needs to sense until finding a channel free of PUs and other nodes, to represent the spectrum sensing performance. Therefore, we define the basis for core update as:

Definition 3.4. For a node u and its core v ($v = c_u$), u needs to update c_u if and only if $A_{uv} > A_u$, where A_{uv} is the estimated average number of channels to sense if u receives assistance from v ; A_u is the estimated average number of channels to sense if u senses itself u and gains no assistance from others.

However, for a node, it does not know its sensing performance without its core, or A_u , since it always senses based on the channel list provided by the core. Another question arises here: how can a node know if its sensing performance is increased or decreased by the core? Obviously, it is difficult for a node to evaluate the assistance by its core. But, from the opposite direction, a core can evaluate its assistance

to its member. Our basic idea is: when a node pushes back the updated channel information, the core evaluates the assistance it could give to the node, under the virtual situation that if the node sends a request now, rather than pushing back its current channel information. The detailed process is:

1. When a node u pushes its newly updated channel information to its core, v ($v = c_u$), v calculates the $L_v^0(M)$ without using the new information from u ;
2. Based on the new channel information from u , v calculates the new $L_v^1(M)$, finds the channel in $L_v^1(M)$ that is recently updated as available, and also has the minimum index in $L_v^0(M)$, which equals to the estimation of A_{uv} ;
3. Then, v calculates A_u , that takes to u to sense randomly until finding a channel free from PUs and other nodes. The current channel availabilities are assumed to be consistent with the channel information in $L_v^1(M)$;
4. Core v compares A_{uv} and A_u . If $A_{uv} > A_u$, then we claim that the assistance from v does not increase the sensing performance of u .

In Step (1), $L_v^0(M)$ is actually the channel list that v would send to u without using any u 's new information, if u sends a help request to v now. In Step (2), the new $L_v^1(M)$ denotes the real channel availability currently on node u . In Step (3), based on the information in $L_v^1(M)$, v can know the channel situation on u . Then, A_u is the expected number of sensing if u randomly picks a channel to sense, which can be simply calculated using basic probability theory. In Step (4), v compares A_{uv} and A_u to see if u 's sensing performance can be improved by v 's assistance, compared to its random-sensing performance.

For example, in Fig. 3.7, suppose the current 3-tuples on node v associated with the three channels are: channel 1, $\langle 13, 8, 9 \rangle$; channel 2, $\langle 11, 10, 7 \rangle$; channel 3, $\langle 5, 6, 12 \rangle$. Now u sends its updated channel information to v , and the new

channel 3-tuples are : channel 1, $\langle 13, 8, 16 \rangle$; channel 2, $\langle 20, 10, 7 \rangle$; channel 3, $\langle 5, 6, 12 \rangle$. Then, v calculates $L_v^0(M) = \{1, 2, 3\}$, and $L_v^1(M) = \{2, 3, 1\}$. The channel that is recently updated as available in $L_v^1(M)$ is channel 2, since $20 = \text{Max}\{20, 10, 7\}$. The index of channel 2 in $L_v^0(M)$ is 2. Therefore, $A_{uv} = 2$.

Since in $L_v^1(M)$, only channel 2 is available, the expected number to find an available channel if u randomly picks a channel to sense is: $A_u = 1 \times \frac{1}{3} + 2 \times \frac{2}{3} \times \frac{1}{2} + 3 \times \frac{2}{3} \times \frac{1}{2} \times 1 = \frac{7}{6}$. In this example, $A_{uv} > A_u$, which means u can have a better performance if it randomly picks a channel for sensing without v 's channel list. Therefore, based on Definition 3.4, u has the need to designate a new core.

When a core node finds that one of its members needs to designate a new core, it would send the update notification to that node after the phase *Push*. Then, the notified node needs to find a new core.

One important benefit with the core structure is that it is easier to propagate, compared to traditional cluster structures. Suppose node u has the necessity to find a new c_u ; the process is very efficient, as described here:

1. u replaces the input N_u of Algorithm 2 with $N_u - \{v\}$;
2. u reruns Algorithm 2, and the output is the new c_u ;
3. u sends a notification to inform the new c_u .

The above process shows that only the node u and the new c_u are affected, and no other nodes are involved. Therefore, the core evolution here takes a very limited cost.

3.2.5 Mixed Cluster-core Structure

Although the core structure is very convenient to construct and update, it is possible that in a sparse network, the core structure cannot assist too much during the spectrum sensing. For example, many nodes are their own cores, whose members are also themselves only. Under these circumstances, we consider having nodes within

more than one-hop distances help each other for spectrum sensing. In this section, we propose a mixed structure of clusters and cores, to involve nodes within multiple-hop distances.

To increase the assistance that a core can provide, we select cluster heads from the cores, and build a cluster structure on top of the core structure. First, for a core u , we use NC_u to denote the set of its neighbor cores. If $v \in NC_u$, then $v \in N_u$ and v is a core with $D_v > 0$. Then, we can define the weight WC_u of a core u , which is later used for cluster head selection:

Definition 3.5. For a core v , the weight of it is defined as $WC_u = \sum_v |M_u \cap M_v|$, $\forall v \in NC_u$.

The definition for core weight is similar to Definition 3.2. The difference is that the weight is calculated based on the core neighbor set, NC_u , instead of the original neighbor set, N_u .

The cluster head selection is based on the weight of each core, and applies the classical cluster construction scheme [50]:

1. All cores are initially uncovered;
2. An uncovered core u becomes a cluster head, denoted as h_u , if it has the highest weight (based on Definition 3.5) among NC_u ;
3. The selected cluster heads and their connected 1-hop neighbor cores are marked as covered;
4. Repeat Steps 2 and 3 on all uncovered cores (if any).

The coverage of the cluster algorithm has been proved in [50]. One example is in Fig. 3.9: we added nodes u' , w' , and x' compared to the example in Fig. 3.7. The newly added three nodes have the same channel information with nodes u , w , and x . Therefore, $c_{u'} = v$, $c_{w'} = u'$, and $c_{x'} = u'$. Now there are three cores, u , v , and u' .

Based on the cluster selection scheme, core v is the cluster head for u , and u' , which means $h_u = v$, $h_{u'} = v$ and h_v is v itself.

Since the cluster-core structure is 2-layered, the spectrum sensing scheme based on it should contain not only the interaction between cores and their members, but also the interaction between cores and their clusters. Here, under the cluster-core structure, the work on the node side remains unchanged as in Section 3.2.3. The work on the core side needs to change to enable the communication with cluster heads.

One option is similar to the scheme in Section 3.2.3, which is that every time the core receives a request, it pulls the latest channel list for the cluster head, and pushes the updates of channel information back to the cluster head. However, since a cluster head usually has more members than a core, using the same scheme as in Section 3.2.3 could potentially cause very large overhead when the request to the cluster head becomes more frequent.

Here, we choose another option, which is to have the cluster head periodically push the updated channel information to the cores. This is a tradeoff between reducing the communication overhead and getting the most recently updated channel list, on cluster heads. For a core v and its cluster head h_v , the process works as follows:

1. The cluster head h_v periodically collects information from the cores in the same cluster, labels each channel m , $m \in M$, with its associated 3-tuple attributes, $\langle TA(m), TN(m), TP(m) \rangle$, and then broadcasts the channel set M with corresponding 3-tuple attributes to all the cores in its cluster;
2. After v gets the information from h_v , v updates its channel information. Next time, when v receives a help request from nodes in D_v , it calculates the channel list $L_v(M)$ based on the newest channel information.

For example, in Fig. 3.9, node v periodically collects information from u and u' ,

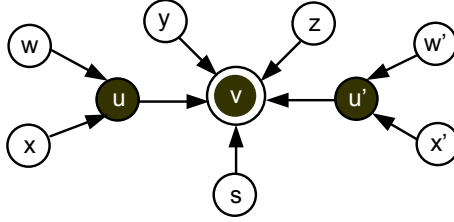


Figure 3.9: Cluster head selection from cores.

and shares the channel information with them. When node w sends a help request to u , u will return the channel list to w . When node w completes data transmission and pushes the updated channel information to u , this update from w would be picked up by v , during the next time v collects information from u . Also, this update will be shared to u' , and assist nodes in $D_{u'}$, which contains v' and w' , for their spectrum sensing.

It is not true to claim that one of the core-only and cluster-core structures is always better than the other. In a dense network, each core in the core-only structure may contain enough members, and is able to provide effective assistance to its members. A cluster head on the cores in this situation is unnecessary, and may cause additional communication costs. In a sparse network, it is possible that cores in the core-only structure do not have enough channel information to provide assistance. Then a cluster-core structure is a better fit. We will show the performance differences in the simulation.

3.2.6 Simulations

We randomly distribute nodes in a 200×200 unit square. The overall simulation settings are listed in Table 3.2. The number of nodes varies for network density control. The network is time-slot based, and each time slot is 1s. We generate data traffic every 10 time slots on a randomly picked set of nodes in the network for broadcasting. For later comparisons about different information exchange frequencies with cores, every single generated data requires 3 time slots to finish transmission by a

Table 3.2: Simulation settings for hitchhiking evaluation.

Number of PUs	10
Number of nodes	[50, 600]
Number of channels	[5, 20]
PU's transmission range	60
Node's transmission range	[40, 50]
Single data task duration	3
Size constraints for cores	[1, 14]
Minimum PU active duration	[5, 15]
Information exchange frequency for cores	[1, 3]
Information exchange frequency for clusters	[3, 9]

single node. Each PU occupies one certain channel when it is active. Since each PU's transmission is usually longer than that of other nodes, we set the minimum number of time slots for an active PU longer than the node's average data transmission time. At the beginning of each time slot, the PUs are randomly selected to be active for a generated time duration. If a selected PU is already active, then its status and remaining active time slots would not be changed.

We study the influence of different algorithm parameters, based on our discussions in previous sessions, which are: size constraints of the core-only structures, information exchange frequencies between a node and its core, and comparison between cluster-core structures and core-only structures. Based on the objective function, Eq. 3.2-5, the performance we compare is the average number of channels that a node needs to sense in each sensing phase. We also implement a random sensing scheme for better comparison, in which each node randomly picks a channel for sensing every time.

The comparison results of core-only structures with different size constraints are presented in Fig. 3.10. We compare the two size constraints, [1, 10] and [4, 14] with the random sensing scheme. In Fig. 3.10(a), the average number of channels to sense in the random scheme increases when there are more nodes. In the core-only structures, the number of sensing first decreases and then increases. This is because, at first,

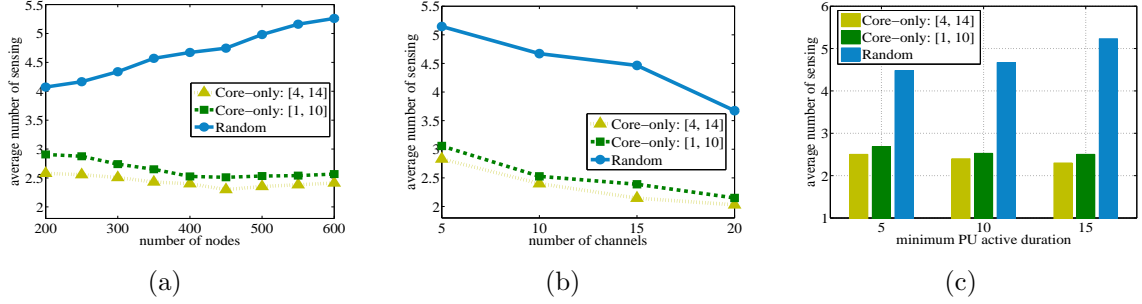


Figure 3.10: Comparison of core-only scheme with different size constraints and random sensing scheme.

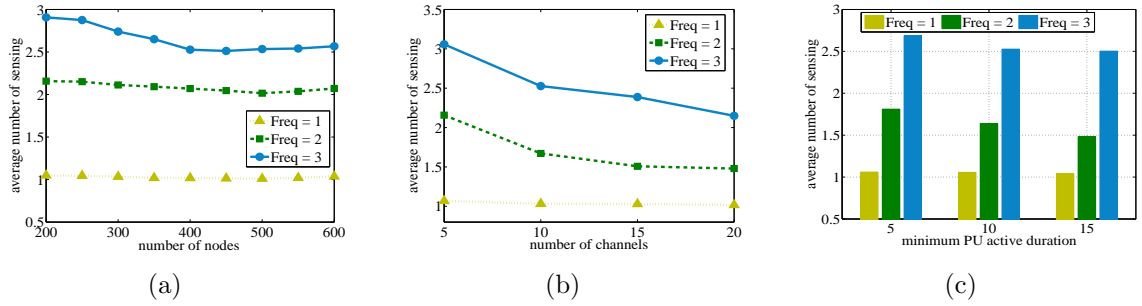


Figure 3.11: Comparison of core-only scheme under different information exchange frequencies.

when more nodes need to use channels, the channel lists on the cores get updated more frequently and are more accurate. However, when the number of nodes reaches a certain amount, the channel availabilities become much less, and each node senses more channels to find an available one. In Fig. 3.10(b), the number of sensing in all three lines decreases when there become more total channels. In Fig. 3.10(c), when a PU's minimum active duration increases, the number of sensing increases for the random scheme. However, both core-only structures decrease. This is because, when a PU occupies its channel for a longer time, the channel availabilities are more static.

We vary the information exchange frequencies between cores and their corresponding members from every 1 time slot to every 3 time slots. The settings of the three network parameters are similar as in the above part. The core size constraints here are [1, 10]. The results are shown in Fig 3.11. The trends of the three frequency settings

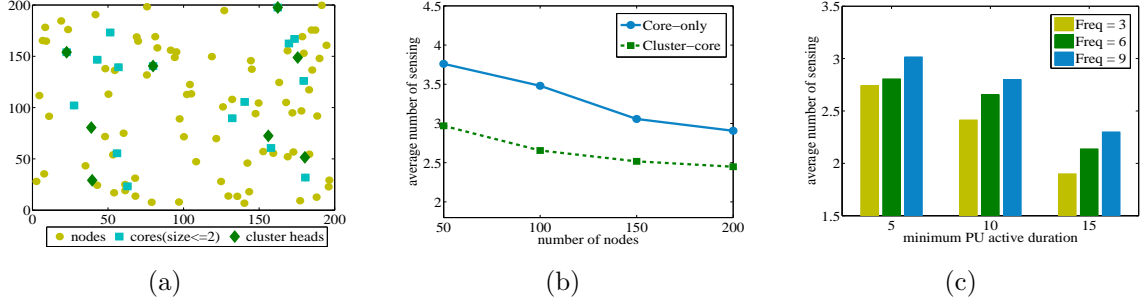


Figure 3.12: Construction and performance comparison of cluster-core scheme.

are similar. And the number of sensing for frequency 3 is the largest among all three, while that for frequency 1 is the least. However, setting the frequency as 1 requires the information exchange with the core every time slot, which is very expensive in a real-life scenario. In addition, the results are more stable when the information exchange frequency is 1. This is because each node can know the instant channel list on its core, which is very accurate, and less influenced by different network parameter settings.

The mixed cluster-core structure is applied when facing sparse networks. Therefore, we set the number of nodes in the network from 50 to 200. The size constraints for selecting cores are $[1, 10]$, and the information exchange frequency between cores and their members is 3. We first show the results of selecting cluster heads from cores in Fig. 3.12(a). The number of nodes in Fig. 3.12(a) is 100. We mark partial cores, whose sizes are less than or equal to 2, which means the core structures for those nodes are not very helpful. Then the cluster heads are marked out using green diamonds. The number of cluster heads is 8 in this case.

In Fig 3.12(b), the performances of the core-only structure and the cluster-core structure are compared. We keep the information exchange frequencies between cluster heads and cores as 6. Since the comparisons of the two schemes by varying other network parameters are similar, we only show the results of changing network densities here. We vary the number of nodes from 50 to 200. The cluster-core structure

requires less number of sensing than the core-only structures. In addition, as the number of nodes increases, the gap between the two schemes reduces.

In Fig 3.12(c), we vary the frequency of information exchange between cluster heads and cores from every 3 time slots to every 9 time slots. The number of nodes is set to 100, and the minimum PU active duration changes from 5 to 15. From the results, we can see that, the more frequently that the cluster head collects information and sends to the cores, the less sensing there is. Moreover, for all three bars, when the average time that a PU takes a channel is longer, the less number of sensing both schemes need. This is caused for a similar reason as in core-only structures, which is when a PU occupies a channel for a longer time, the channel availabilities are less dynamic, and the cluster-core structure can provide more accurate information.

3.2.7 Conclusion

In this section, we propose two frameworks to assist nodes in CRNs during spectrum sensing. One is the core-only structure based, and the other is the cluster-core structure based. We describe the construction of the core-only structure, and the spectrum sensing scheme assisted by it. Also, we consider the evolutions of cores by taking the channel dynamics in CRNs into account. Our core structures take very limited propagation costs. In the cluster-core structure, we describe the application scenarios and how to select cluster heads from current cores. The sensing scheme assisted by the core-cluster structure is discussed. We perform numerous simulations to testify to the performance of our frameworks, and study the influences of different parameter settings.

3.3 Chapter Conclusion

In this chapter, we focus on how to improve the spectrum sensing performance while reducing the time cost in CRNs. We study the pre-phase for spectrum sensing,

which is how to select spectrum channels for sensing. This is the part that has been neglected in previous works. By choosing channels that are more likely to be available, the spectrum sensing performance is increased and the time cost is reduced.

We first propose a sense-in-order (SIO) model, which determines the order of selecting channels to sense for each node, based on the information from both space and time dimensions. It can reduce the delay and energy cost during the spectrum sensing phase. Through the cooperation among SUs, we are able to construct the state transition graph. We identify the possible states for every channel on each node and define the transition among the states, based on the signal that the node receives and the valid time period. When each node needs to perform the spectrum sensing, it can identify states of different channels and choose the ones that more likely to be available to sense first. We later improve the SIO model by taking the angle dimension into account.

Secondly, we propose two structures for the sensing assistance to assist each node in hitchhiking in its spectrum sensing. The first one is the core-only structure assisted, in which each node designates a neighbor or itself as its core, and can gain help from its core during the spectrum sensing phase. Given the dynamic channel situations, we also propose an evolution model for each node to adjust its core selections, and seek the core that can provide the most help. In addition, we consider that, in a sparse network, the number of nodes designating a same core is very small. This results in a core-only structure unable to provide enough assistance for spectrum sensing. Therefore, we provide a 2-layer structure of both clusters and cores, and a 2-layer spectrum sensing scheme to fit the mixed cluster-core structure.

In the next chapter, we will discuss the channel assignment problems in CRNs. Different from the spectrum sensing phase, channel assignment is very important for MAC and routing layer protocols.

CHAPTER 4

CHANNEL ASSIGNMENT IN CRNS

Since the current static allocation of channels in traditional wireless networks is inefficient for CRNs, this chapter presents our work on the channel assignment in CRNs. We first consider the pure channel assignment which takes the channel dynamics into account. Then, we present the channel assignment framework under the dynamic source routing protocol.

4.1 Effective Channel Assignments

The channel assignment problem is well-studied in traditional wireless networks, with the objective of satisfying the interference constraints and maximizing the number of nodes with channels assigned. In its most general form, the channel assignment problem is equivalent to the generalized graph-coloring problem, which is a well-known NP-hard problem [51]. An extensive survey of channel assignment in wireless networks, including CRNs, can be found in [4].

The fundamental difference between CRNs and traditional wireless networks is that the available channel sets are dynamic, and their availabilities vary over time and space. In CRNs, the channel assignment problem has been studied from different perspectives. Some of them aim to maximize the spectrum utilization [52], subject to interference constraints. Some other works study the cross-layer optimization, including using power control [53, 15], and considering both network and link layers [54]. Our focus here is channel assignment at the link layer only.

In this section, we first present two basic localized algorithms and an advanced

localized algorithm to solve the channel assignment problem. Based on the proposed framework, we further develop three conflict resolution strategies to make our scheme versatily adaptable to different network conditions.

4.1.1 Preliminaries

We consider a CRN as a graph $G = (V, E)$, where $V = \{u, v, w, \dots\}$ is the node set representing wireless nodes, and E is the link set. $uv \in E$ if nodes u and v can communicate with each other. $N(u)$ represents the neighbor set of node u . We assume that the whole spectrum is divided into multiple channels of different central frequencies. To simplify our model, we treat the performance (e.g., bandwidth) of each channel as the same. Our model can be extended to situations, where the bandwidth of each channel is different, by adding the channel performance as the weight to each link. $C_u \subseteq C$ denotes the set of available channels on u , where C denotes the set of total available channels in the network. C_u is not equal with C , due to the different interference ranges of primary users at different locations. We also assume that there is a CCC for nodes to exchange information.

In wireless networks, two adjacent nodes can communicate only when they both tune to the same channel. A link exists when two adjacent nodes select the same channel. Two links are adjacent if they share one end node. Conflicts exist if two adjacent links are assigned the same channel. The goal of our paper is to perform channel assignment so that the maximum number of links exist without conflicts.

We make the following assumptions used in the paper:

1. The communication range equals the interference range, to make our algorithms and analysis more clear and concise. Our model can be extended to more sophisticated ones, as shown in our simulation.
2. Each link is only assigned with a single channel.

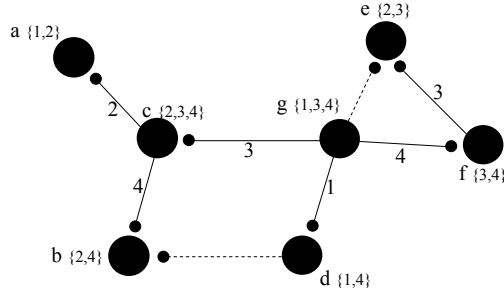


Figure 4.1: The example topology for channel assignment.

We use the topology of Fig. 4.1 to illustrate our algorithms. The available channel set on the whole network is $C = \{1, 2, 3, 4\}$. The node set is $V = \{a, b, c, d, e, f, g\}$ and the link set E is the edge set, shown in Fig. 4.1. (The link connections and link labels will be described later.) Suppose there is a certain number of primary users in the network, which results in different available channel sets among nodes, as shown in Fig. 4.1. For example, the available channel set on node a is $\{1, 2\}$. This is because node a is within the interference range of primary users occupying channels 3 and 4. The available channel set on node c is $\{2, 3, 4\}$ since it is within the interference range of primary users using channel 1.

4.1.2 Basic Local Algorithms

Two basic algorithms are described: one is the node-based algorithm without coordinations between adjacent nodes, and another is the link-based algorithm with coordinations.

The node-based algorithm, which uses only local channel information at each node to select channels for adjacent links, is given in Algorithm 4. It randomly assigns channels for each link, based on the information of each node. There is no coordination between two end nodes of one link. Obviously, its efficiency is low. The probability of selecting the same channel at both end nodes of a link is small, which results in many rounds being needed to complete the algorithm.

Algorithm 4 Node-based selection

```
1: {Initial allocation phase}
2: for  $\forall v \in V$  do
3:   for  $\forall u$  such that  $vu \in E'$  do
4:     if  $|A_{vu}| = 0$  and  $|C'_v| > 0$  then
5:       randomly pick  $c$  from  $C'_v$ 
6:        $A_{vu} \leftarrow c$ 
7: {Conflict resolution phase}
8: for  $\forall uv \in E'$  do
9:   if  $a_{uv} = a_{vu}$  then
10:     $A_{uv} \leftarrow \{a_{uv}\}$ ,  $E' \leftarrow E' - \{uv\}$ 
11:     $C'_u \leftarrow C'_u - \{a_{uv}\}$ ,  $C'_v \leftarrow C'_v - \{a_{uv}\}$ 
12: if  $\exists mn$  s.t.  $A_{mn} = 0$  and  $(|C'_m| > 0$  or  $|C'_n| > 0)$  then
13:   go to step 1
```

The link-based algorithm is described in Algorithm 5. We first give the definition of admissible channels, which will also be used later.

Definition 4.1. The admissible channels for link uv is defined as $C_u \cap C_v$, denoted as C_{uv} .

To simplify our discussion, we exchange the role of nodes and links. In this case, uv corresponds to a node. uv 's neighbors are either uw or vw . After this exchange, adjacent links become adjacent nodes. Then, we focus on channel selections for nodes instead of links. So the nodes in the algorithm description below are actually the original links. This is the conflict graph construction process, which can also be found in [55].

Unlike the node-based solution, the link-based solution will result in conflicts among adjacent links (new nodes). Local solutions vary depending on how (1) admissible channels are selected and (2) conflicts among adjacent nodes are resolved. These methods can be based on either a random choice or a predefined priority. The simple approach in Algorithm 5 is to have a random admissible channel selection from C_{uv} and conflict resolution based on node id: $ID(uv) = ID(u) + ID(v)$. That is, node uv with the highest $ID(uv)$ will win.

Algorithm 5 Link-based selection

```
1: {Initial allocation phase}
2: for  $\forall vu \in E'$  do
3:   randomly select  $c$  from  $C_{vu}$ 
4:    $A_{vu} \leftarrow c$ 
5: {Conflict resolution phase}
6: for  $\forall uv \in E'$  do
7:   if  $uv$  and any link in  $N_{uv}$  have conflicts then
8:     remove the channel from the link with the lowest priority
9:   if  $A_{uv} > 0$  then
10:     $E' \leftarrow E' - \{uv\}$ ,  $C_{uv} \leftarrow C_{uv} - A_{uv}$ 
11: if  $\exists mn$  s.t.  $A_{mn} = 0$  and  $C_{mn} > 0$  then
12:   go to step 1
```

Algorithm 5 reduces the number of rounds needed by channel assignment compared to Algorithm 4, since there is a coordination between two end nodes during channel selection. The coordination here is achieved by two end nodes exchanging their available channel information through the common control channel. However, Algorithm 5 does not take priorities of different links into consideration; this would still result in a relatively low efficiency. In the next section, we will present an advanced algorithm, which considers the priorities of links and applies maximal matching.

4.1.3 Advanced Local Algorithm

Different from the previous two basic algorithms, the node-link-based algorithm makes improvements on both the sides of initial assignment and conflict resolution.

For the initialization part, we propose a notion of “star,” given as follows:

Definition 4.2. A star is a special 2-level tree with one node, and a set of adjacent links associated with that node.

In each star, each link is “handled” by the end node, called a *host*. The node with a higher ID is the host. This process is called a ‘partition based on node ID.’ In this way, each link is associated with one node that has a larger ID. This partition will

form a forest of “stars.” Then, in each “star,” it is possible to perform a good initial assignment through maximal matching processing, by assigning channels to links that minimize channel-conflict probability. This will maximize channel weight, which is defined later.

Suppose a link uv , $uv \in s_u$, selects a channel $c \in C$, the *conflict probability*, with its neighbors, is depicted as the following modified $|C_{uv}|$:

$$p_{uv}(c) = \frac{T_{uv}(c)}{\sum_{uv \in s_u} \sum_{c' \in C} T_{uv}c'} \quad (4.1-1)$$

where

$$T_{uv}(c) = \sum_{w \in N_v} \frac{1}{|C_{uw}|} E_{uw}(c) + \sum_{w \in N_u} \frac{1}{|C_{vw}|} E_{vw}(c),$$

$E_{uw}(c)$ is a step function with a value of 1 when $c \in C_{uw}$, and 0 otherwise. Suppose $d_{uv} = d_u + d_v - 1$, where $d_u = |N_u|$.

We can easily derive that for each star s_u ,

$$\sum_{uv \in s_u} \sum_{c \in C} p_{uv}(c) = 1.$$

Based on the definition of conflict probability, we define the following channel weight:

Definition 4.3. The channel weight of channel c over link uv is defined as

$$w_{uv}(c) = d_{uv} \times (1 - p_{uv}(c)).$$

For the conflict resolution part, we propose a local and greedy solution by considering various priorities, which are related to the importance of each node in resolving conflicts. Let d_u be the *effective node degree* of node u , defined as the total number of neighbors minus the number of neighbors with channels assigned. The importance of

Algorithm 6 Node-link-based selection

```
1: {Initial allocation phase}
2:  $S \leftarrow$  partitions of  $G$  according to ID
3: for  $\forall s_v \in S$  do
4:   for  $\forall vu \in s_v, \forall c \in C_{vu}$  do
5:     calculate  $w_{vu}(c)$ 
6:   calculate maximal matching between channels and adjacent links by the Hun-
   garian's algorithm
7:   for  $\forall vu \in s_v$  do
8:     update  $A_{vu}$ 
9: {Conflict resolution phase}
10:  $\mathbf{E}' = \mathbf{E}$ 
11: for  $\forall uv \in E'$  do
12:   if  $uv$  and any link in  $N_{uv}$  have conflicts then
13:     remove the channel from the link with a lower effective degree
14: for  $\forall s_v \in S$  do
15:   for  $\forall vu \in s_v$  do
16:     if  $A_{vu} > 0$  then
17:        $s_v \leftarrow s_v - \{vu\}$ 
18:        $E' \leftarrow E' - \{vu\}, C_{vu} \leftarrow C_{vu} - A_{vu}$ 
19: if  $\forall s_v$  satisfies  $|C'_v| > 0$  and  $A_{vu} = 0$  for  $\forall vu \in s_v$  then
20:   go to step 3
```

a node is then defined as its effective node degree. This strategy will establish more connections quickly, since the node with more links has a higher priority.

Combining both processes introduced above, we can give Algorithm 6 the *node-link-based* selection algorithm. Suppose the host of link uv is u . u needs to collect C_u, C_v , and C_w for all $w \in N_v \cup N_u$ to calculate channel weight for uv . Therefore, two-hop information is needed (i.e., a neighbor's neighbors). This process can be done through two rounds of exchanges, using a common channel. Step 1 of Algorithm 6 requires one round of exchanges, and is calculated only once. Step 5 needs to be re-calculated at each round, as G changes at steps 15, 16, and 17.

The maximum matching is done through constructing a bipartite graph with channels at the left side, and adjacent links at the right side. The weight value of each mapping edge is the corresponding channel weight on the link. We apply the Hungarian's algorithm[56] here to find the maximum matching, which can be completed in

polynomial time, $O((|C|\Delta)^4)$, where $|C|\Delta$ is the maximal number of links in each bipartite graph. Moreover, the Hungarian's algorithm is a local greedy algorithm, since it only needs the weight information on each adjacent link in one star. The number of channels and the number of links can be made equal by adding virtual nodes at either side, so that the number of channels and the number of links are the same. To apply perfect matching, the bipartite must satisfy *Hall's matching theorem*[56] by adding virtual edges from the virtual nodes. This would not affect the final result.

Theorem 4.4. *The adding of virtual node and virtual edges would not affect the result of perfect matching achieved by the Hungarian's algorithm.*

Proof: Given a bipartite graph $G = (V = (L, R), E)$, such that for each $(u, v) \in E$, $u \in L$ and $v \in R$. Before adding virtual nodes and virtual edges, suppose the maximal perfect match achieved by the Hungarian's algorithm is $M = (V, E')$, $|L| = |R|$ and $E' \subset E$. In addition, for each $u \in L(v \in R)$, there is exactly one $v \in R(u \in L)$, such that $(u, v) \in E'$. The maximal perfect match after adding virtual nodes and virtual edges is $M' = (V', E'')$. Let $M_0 \subset M'$, such that $M_0 = (V_0, E_0)$ is composed of virtual edges whose weight are 0, where $V_0 \subset V'$ are virtual nodes and $E_0 \subset E''$ are virtual edges.

First, assume the weight of all edges in M , $W(M)$, is less than the weight of all edges in M' , $W(M')$. We exclude all the virtual edges from E'' and obtain $W(M' - M_0)$. $W(M' - M_0) = W(M') - W(M_0) = W(M')$ because $W(M_0) = 0$. This means that $W(M) < W(M' - M_0)$ with $M' - M_0 \subseteq G$. That is, $M' - M_0$ can also be the perfect matching of G , which contradicts that M is the maximal matching, since $M = (V, E')$ is the output of the Hungarian's algorithm.

On another hand, assume that $W(M) > W(M')$. Then, $M = (V, E')$ can also be a matching in the bipartite graph with virtual nodes and virtual edges. This contradicts the fact that $M' = (V', E'')$ is the maximal matching of the bipartite graph with virtual nodes and virtual edges, which should be the maximal. Since

virtual edges do not influence the final result, virtual nodes do not affect either, because edges coming from virtual nodes are virtual edges with weight 0. \square

In step 9, resolving conflicts requires exchanges among host nodes, which correspond to two rounds of exchanges. There are several ways to resolve conflicts through priority, which we will discuss in details later. One priority is the combined effective node degree of two end nodes. Another priority is based on channel weight $w_{uv}(c)$. The higher the weight, the higher the priority. $|C_{uv}|$ can also be used as a priority, with a small value corresponding to a higher priority. Removing assigned links in steps 10 and 11 requires only local operations in hosts.

We now give a specific example based on Algorithm 6 to better illustrate our algorithm. Considering the topology in Fig. 4.1, suppose that $ID(a) = 1, ID(b) = 2, \dots, ID(g) = 7$. The admissible channel set for each link is shown in Table 4.1. The original graph is partitioned into three stars, as shown in Fig. 4.1, in which links are only connected with nodes in stars. The three stars are: c with its attached links, g with its attached links, and f with its attached links. We compute the conflict probability of every available channel on each link, which is in Table 4.2. Next, we can get the weight of each channel on each link. The results are in Table 4.3.

Here, we take the channel assignment on the star charged by node c for an example. We construct a bipartite graph, and add two virtual nodes on the link side to conduct the perfect matching. Each edge in the bipartite graph has a weight, as computed in Table 4.3. The weight of virtual edges connecting virtual nodes is 0. Next, we conduct the maximum matching shown in Fig. 4.2. The other three stars conduct their channel assignments in the same way. The total view of channel assignment results is in Fig. 4.1. The number on each link is the channel assigned to it. Since link ge cannot get any channel, we use a dotted line to represent this link.

There is a conflict between links cb and db , because they both are assigned channel 4. We resolve this conflict based on the effective node degree of two end nodes, which

Table 4.1: Admissible channel set on each link.

ca	cb	gc	gd	ge	gf	db	fe
2	2, 4	3, 4	1, 4	3	3, 4	4	3

Table 4.2: Conflict probability of every channel on each link.

$p_{ca}(2)$	$p_{cb}(2)$	$p_{cb}(4)$	$p_{gc}(3)$	$p_{gc}(4)$	$p_{gd}(1)$
$\frac{4}{13}$	$\frac{4}{13}$	$\frac{5}{13}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{3}{16}$
$p_{gd}(4)$	$p_{ge}(3)$	$p_{gf}(3)$	$p_{gf}(4)$	$p_{db}(4)$	$p_{fe}(3)$
$\frac{1}{16}$	$\frac{1}{8}$	$\frac{3}{16}$	$\frac{3}{16}$	1	0

Table 4.3: Weight of every channel on each link.

$w_{ca}(2)$	$w_{cb}(2)$	$w_{cb}(4)$	$w_{gc}(3)$	$w_{gc}(4)$	$w_{gd}(1)$
$\frac{27}{13}$	$\frac{36}{13}$	$\frac{32}{13}$	$\frac{21}{4}$	$\frac{21}{4}$	$\frac{65}{16}$
$w_{gd}(4)$	$w_{ge}(3)$	$w_{gf}(3)$	$w_{gf}(4)$	$w_{db}(4)$	$w_{fe}(3)$
$\frac{75}{16}$	$\frac{35}{8}$	$\frac{65}{16}$	$\frac{65}{16}$	0	3

turns out to be the same. Therefore, we remove one randomly. During the next round of our algorithm, the result is not changed. Thus, the channel assignment process is completed.

4.1.4 Improved Conflict Resolutions

In the above discussion, we use the effective degree for conflict resolution. There are several other ways to resolve conflicts through priority. In this section, we propose another two conflict resolutions.

The first alternative is based on the remaining number of channels. For example, $|C_{vw}|$, which is the number of unused channels on vw , can also be used as a priority, with a smaller value corresponding to a higher priority. This strategy assigns a higher priority to links with fewer choices of channels.

The second alternative is based on whether the link is *essential* or *nonessential*. First, we assume that the priority of each node i is $ID(i)$, based on alphabetical order, such as $ID(a) < ID(b)$. Here, for simplification, the priority is decided by nodes' IDs. In real life, other meaningful metrics, for example, bandwidth and capacity, can

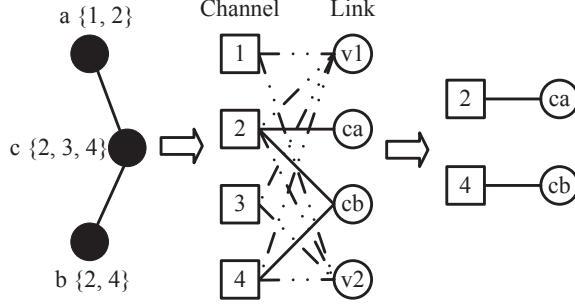


Figure 4.2: The channel assignment process of the star charged by node c .

be applied to decide the priority. Then, we introduce the priority assignment method for each link [57].

Definition 4.5. Link priority assignment: For each link uv , its priority is defined as $Pr_{uv} = (ID(u), ID(v))$.

Thus, the priority of a link is a 2-tuple, which is based on the lexicographic order. To decide the priority, first compare the first element in the 2-tuple, and then compare the second element. The first element of the 2-tuple is the priority of the start node, and the second element is the priority of the end node. Therefore, each link has a total order in the network. Next, we give the definition of a nonessential link, based on the link priority.

Definition 4.6. A link uv is a nonessential link if it satisfies the following conditions:

- (1) There is a “replacement” path P from u to v , which does not pass link uv , and
- (2) all the intermediate links on P have higher priorities than link uv .

Any link that does not satisfy the above condition is essential. For example, suppose that there is a path from u to v , which is $u \rightarrow w \rightarrow v$. Then, link uv is nonessential, because the intermediate links uw and wv both have higher priorities than link uv .

Therefore, when a conflict happens among two adjacent links, we can first check if they are essential. The nonessential ones will be removed directly. If the adjacent

Table 4.4: Simulation settings for local channel assignment algorithms.

total number of nodes	[10, 40]
communication range of each node	[50, 70]
total number of channels	[4, 30]
total number of PUs	10
interference range of PUs	[40, 140]

links are both essential, we will use other methods - effective node degree, or the number of channels remained on the link - to decide which one should be removed.

4.1.5 Simulations

Here, we present simulation results for our three algorithms. In addition, we implement two other algorithms: the distributed greedy algorithm in [19] and an optimal algorithm for comparison, which are compared with our node-link-based algorithm. The greedy algorithm iteratively assigns channels to the node with minimum channel choices. Our algorithm does not need the iterative process. We calculate the maximal matching among links and channels, and then resolve the conflicts. Also, we compare the three conflict resolutions of our third algorithm. Moreover, we implement the approach proposed in our discussion part, and make some comparisons.

We randomly distribute nodes in a 200×200 unit square. Also, we randomly generate a certain number of nodes and PUs with different interference ranges. Each primary user occupies one channel in a certain range. If a node is within a PU's range, it cannot use the channel occupied by the PU. Therefore, each node in the network has its own available channel set, according to the positions of PUs. The settings of parameters are shown in Table 4.4. Under a certain setting, we run our simulation 100 times to achieve stable results.

The three parameters, total number of nodes, total number of channels, and interference range of primary users, are tunable. Each time, we change one of the parameters to compare the algorithms using the following metrics:

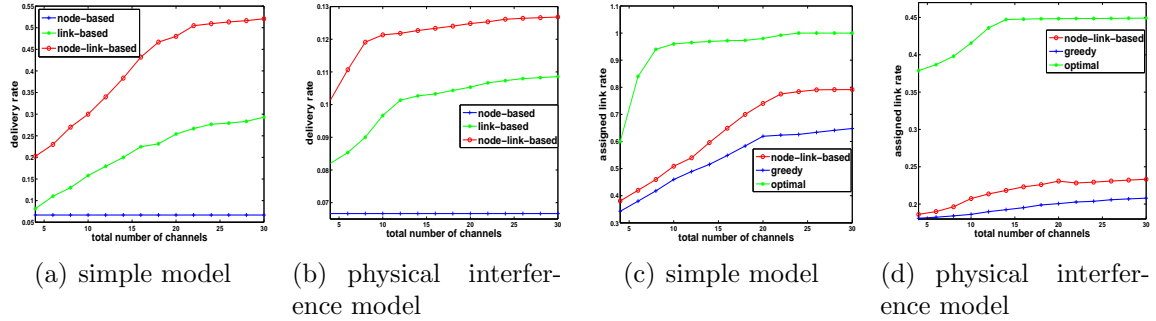


Figure 4.3: Comparison of delivery rate in two models.

- *assigned link rate*: the ratio of assigned links over possible links,
- *delivery rate*: the ratio of the maximum broadcast reachable nodes over total number of nodes, and
- *number of rounds*: the number of rounds needed by channel assignment.

The higher the assigned link rate, the better the result. The same applies for the delivery rate. A small number of rounds indicates higher efficiency. Based on the above system settings and evaluation metrics, the simulation results are presented in the next section.

We assume that the interference range equals the communication range. Because of the accumulative effect of interference, the physical interference model (i.e. the SINR model) is generally considered as a more realistic model. [58] provided a per-node interference range calculation method, which performs very closely to the physical interference model.

Our design could be easily extended to apply the model in [58]. We vary the number of channels from 4 to 30, while keeping the number of nodes at 15 and interference range of PUs at [60, 70]. The interference range of each node is computed separately for the physical driven model. We compare the delivery rate of our three algorithms before and after applying this model. Fig. 4.3(a) and Fig. 4.3(b) have similar comparison results. In both figures, the performance of the node-link-based

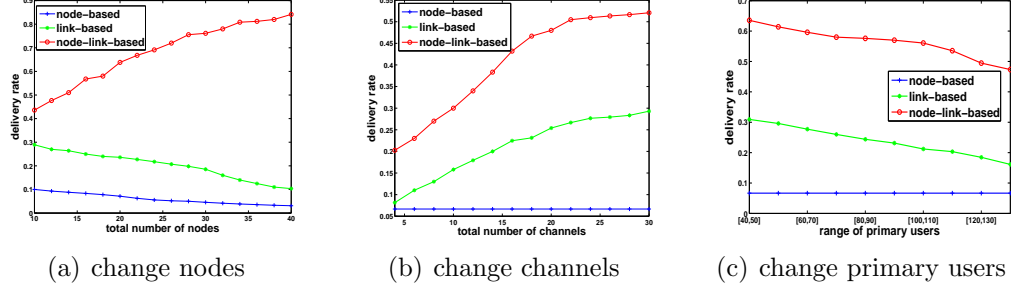


Figure 4.4: Comparison of delivery rate among three proposed algorithms.

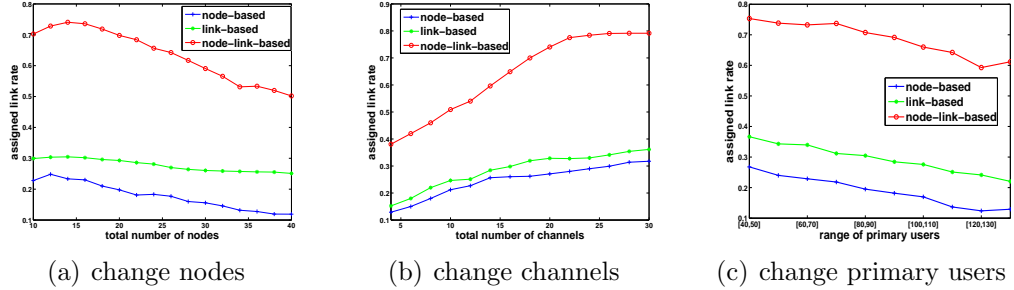


Figure 4.5: Comparison of assigned link rate among three proposed algorithms.

algorithm is the best, and the node-based algorithm is the worst. In addition, we implement two other algorithms. One is an optimal algorithm which maximizes the assigned link rate, without consideration of the number of rounds it takes. Another one is the distributed greedy algorithm in [19]. We compare the assigned link rate of the two algorithms and our node-link-based algorithm. Comparison results in Fig. 4.3(c) and Fig. 4.3(d) show that the comparison results are the same, with similar trends for each algorithm. Comparisons of other metrics are also similar. Therefore, in subsequent simulations, we only consider the simple model.

First, we compare the delivery rate by changing the three parameters. In Fig. 4.4(a), we vary the number of nodes from 10 and 40, while keeping the total number of channels as 10. The interference range of primary users is randomly generated among [60, 70]. In Fig. 4.4(b), we vary the number of channels from 4 to 30, while keeping the number of nodes at 15, and interference range of PUs at [60, 70]. In Fig. 4.4(c), we vary the interference range of primary users, while keeping the number of nodes at 15, and the number of channels at 10. Based on the settings of the three

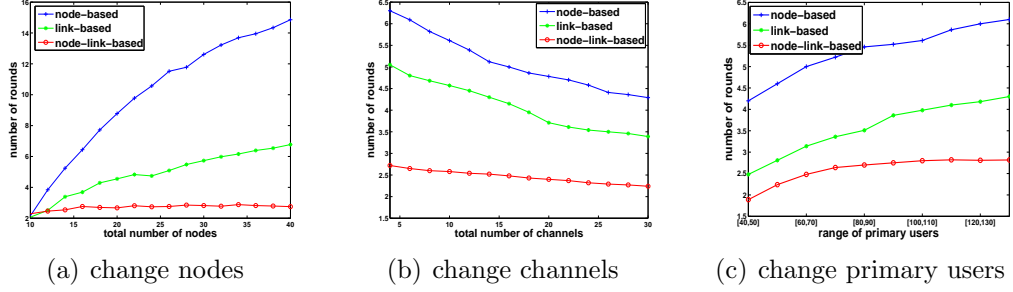


Figure 4.6: Comparison of rounds among three proposed algorithms.

parameters, we randomly generate a topology each time. The results of Fig. 4.4 show that the node-link-based algorithm is almost 50% more than others. The trends of the three vary, because more nodes and larger primary user ranges cause more conflicts, while more channels cause fewer conflicts. The trend of the node-link-based algorithm in Fig. 4.4(a) is increasing because, even though more conflicts cause fewer assigned links, the assigned links are more connected; this causes the increase in the delivery rate.

Second, we compare the assigned link rate. The settings are the same as the ones comparing the delivery rate. The results are shown in Fig. 4.5. The node-link-based algorithm has almost 2.5 times the other two in the assigned link rate. Reasons of the trends are the same as the above argument.

Finally, we compare the number of rounds the three algorithms need to complete channel assignment. The settings are the same, with results shown in Fig. 4.6. The node-link-based algorithm needs the least number of rounds to complete channel assignment, which is always less than three, based on our simulation.

For better comparison, we compare the node-link-based algorithm with the distributed greedy algorithm and optimal algorithm. We present the comparison results in the following:

From the above comparison results, we can find that the two metrics, delivery rate, and assigned link rate give similar results. Here, we only compare delivery rate and number of rounds.

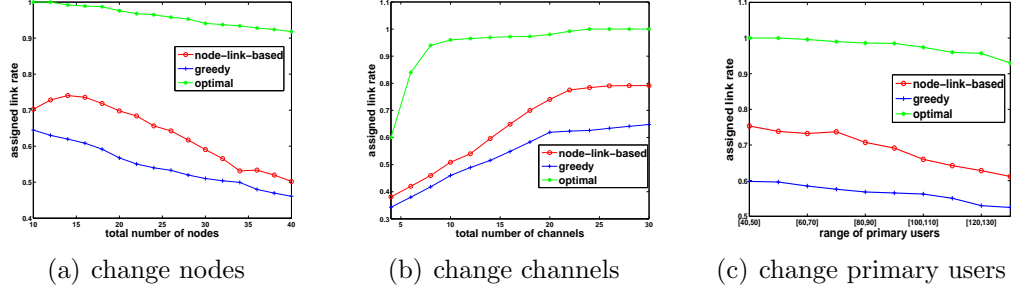


Figure 4.7: Comparison of assigned link rate among node-link-based, greedy, and optimal algorithms.

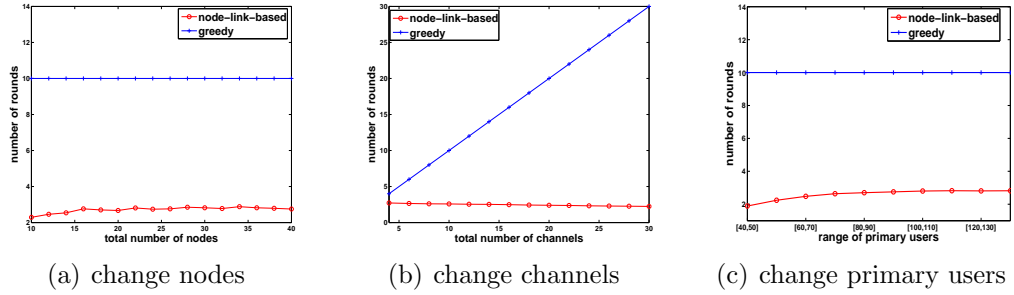


Figure 4.8: Comparison of rounds among node-link-based, greedy, and optimal algorithms.

First, we compare the assigned link rate of the three algorithms. Using the same settings as before, results are shown in Fig. 4.7. We can tell that the node-link-based algorithm achieves almost 70% of the optimal algorithm, and is 10% higher than the distributed greedy algorithm. This is because the optimal algorithm performs channel assignment without consideration of resources.

To better evaluate, we compare the number of rounds needed by the node-link-based and distributed greedy algorithms (the number of rounds is too large in the optimal algorithm). We vary the number of nodes and the number of channels each time. The results in Fig. 4.8 show that the node-link-based algorithm takes less rounds than the distributed greedy algorithm. The number of rounds in the greedy algorithm equals the number of channels. This is because the greedy algorithm needs to run once for each available channel.

In this subsection, we implement the three resolution strategies: effective degree,

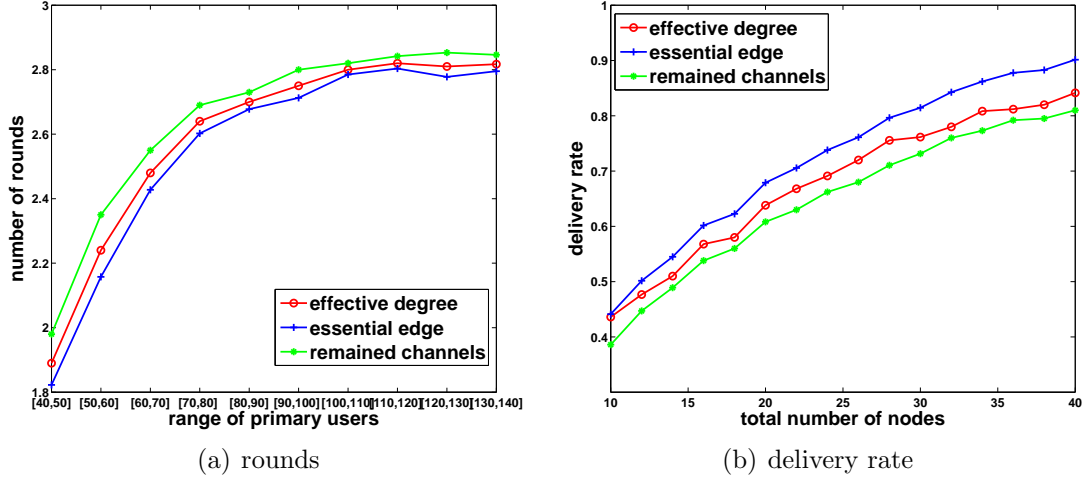


Figure 4.9: Comparison among three conflict resolution strategies.

essential edges, and the number of residual channels. We use the three conflict resolutions to implement the node-link-based algorithm. Under the same settings as before, the results are shown in Fig. 4.9(a). We can see that resolving conflict through essential edges uses a smaller number of rounds than the other two strategies. Also, we compare the delivery rate of the three conflict resolution strategies by changing the number of nodes. The other settings remain the same. The results in Fig. 4.9(b) show that conflict resolution based on essential edges has the highest delivery rate. This is because the replacement paths of nonessential edges do not reduce network

4.1.6 Conclusion

In this section the channel assignment problem in CRNs is studied. We propose three algorithms: node-based, link-based, and node-link-based. In the node-link-based algorithm, we are able to achieve the best localized initialization by using a “star” structure and maximal matching. We also present three ways of conflict resolution. We introduce the notion of essential edges, and propose the possible extensions of our algorithm, with the goals of improving the efficiency while keeping the connectivity of the network. Extensive simulations are conducted to compare our algorithms with others from different aspects. Results show that our advanced

algorithm outperforms existing methods.

4.2 Channel Assignment Under Dynamic Source Routing

In this section, we consider the channel assignment under the scenario where a network is using dynamic source routing (DSR)[59]. Among previous works done on combining routing and channel assignment, [24–26] are about assigning channels among nodes with multiple network interface cards. They do not consider cognitive radio networks and have their own routing scheme. Their approaches have no prediction about nodes' actions and are fixed in the channel assignment process to some extent. In addition, there are mainly two phases in DSR: route detection and route maintenance. We can make use of these phases to gather information about nodes along the route and achieve a more efficient channel assignment scheme.

We first propose a channel assignment approach under a single route model. We use the route reply message (RREP) sent back by the destination node to gather the estimated preference of each node to different channels. The source node can estimate the SINR of each node on a certain channel along the route based on predicting their choices among channels. We give the correctness and complexity analysis of our algorithm. Then, we extend to a multi-route model based on the formation of a multistage graph. We use a locking/unlocking scheme to assign channels for nodes on a multi-route.

4.2.1 Preliminaries

To begin with, we first introduce two preliminaries for our model. One is the SINR model related and another is about the multistage graph.

SINR Model. Works in [60–63] study the physical model for CRNs. In our model, we adopt the SINR model. For a node n working on a certain channel m , the value of $SINR_{n,m}$ in n 's operation area generally follows the following expression:

$$SINR_{n,m} = \frac{S_n}{\sum_{j \neq n} a_{m,j} I_{j,n} + N_0},$$

where S_n is the minimum received signal strength in n_i 's coverage area; $a_{m,j}$ equals 1 when j is using m and 0 otherwise; $I_{j,n}$ is the maximum interference strength a node j can produce to any receiver in n 's coverage area; N_0 is the noise level.

More precisely, $S_n = P_n/Q_{n,n}$, where P_n denotes n 's transmission power, and $Q_{n,n}$ denotes the maximum pathloss from transmitter n to any position in its operation area. $I_{j,n} = P_j/Q_{j,n}$, where $Q_{j,n}$ denotes the smallest pathloss from j to any position in n 's operation area.

The maximum achievable data transmission rate R of a given channel can then be computed based on the given $SINR$ using Shannon's capacity theorem:

$$R = W \log_2(1 + SINR),$$

where W is the carrier bandwidth.

We use the above two expressions in the following sections to estimate the SINR and choose channels based on the data transmission rate.

Multistage Graph. A multistage graph is a graph $G = (V, E)$:

- V is partitioned into $K \geq 2$ disjoint subsets $\{V_1, V_2, \dots, V_K\}$;
- If (a, b) is in E , there exists an i such that either both a and b in V_i or a is in V_i and b is in V_{i+1} ;
- $|V_1| = |V_K| = 1$. The vertex s in V_1 is called the source; the vertex t in V_K is called the sink.

A subset in the multistage graph is called a stage. Detailed examples will be introduced later.

4.2.2 Problem Formulation

We consider a CRN using DSR under the SINR-driven interference model. Each node in the network has a operation area and it must be heard by any point in its operation area. There are two major phases during routing: route discovery and route maintenance. A route reply would only be generated if the message has reached the destination node. Each node in the network is a cognitive radio node and has the ability to choose its own channel.

Each node in our network has its own operation range and it uses a certain amount of transmission power to transmit to nodes in its operation area through one channel. The SINR of any node within a certain node's transmission range should be greater than the SINR threshold. Also, we assume that there is a common channel in the CRN. The source node broadcasts a route request message through the common channel. All of the other nodes also send the probing message on the common channel until the destination node is reached. The destination node would send the route reply message. We use $M = 1, 2, \dots$ to denote the set of available channels and $N = 1, 2, \dots$ to denote the set of nodes on the chosen route. Some nodes cannot use the same channel due to interference. We denote a single node with $n_i \in N$ and a channel $m_i \in M$. After the source node receives the route reply, it would assign $|M|$ channels among $|N|$ nodes on the chosen route. Each node would be assigned to a channel used for transmitting. After performing the channel assignment, the source node would know the throughput of this route. Our approach is a two-stage, best-effort scheme. First, find the best route, in terms of hop counts, to the destination without considering channel conditions and rates. Then, optimally select channels based on rates of the selected path. Clearly, these two stages do not constitute a global optimization, but they represent a good heuristic as is confirmed through our simulation.

In our model, there are several constraints:

- For each node n using channel m along the route, to make sure it can transmit

successfully to any node in its operation area, the SINR of any point in its operation range should be above a threshold, $SINR_{n,m} > \beta$.

- For each node, the transmission rate coming in should equal the transmission rate going out, $f_{n,m'}^{in} = f_{n,m}$.
- For each node, the transmission data rate $f_{n,m}$ cannot exceed the maximal transmission rate $c_{n,m}$ on its assigned channel, $f_{n,m} \leq c_{n,m}$.

Under the constraints, our objective is to maximize the throughput of the selected route. The source node would compute the best channel assignment result based on the information brought back by the route reply.

4.2.3 Single Route Channel Assignment

In this part, we propose a single route channel allocation algorithm under DSR. We would first analyze the hardness of this problem. Then, we would describe two main parts of our algorithm. Finally, we would give correctness and performance analysis.

Hardness Analysis. Given a chosen route, if we do not consider accumulative interference, each node would make its own choice independently. Let f denote the final throughput and c_{n_i,m_i} denote the maximal transmission rate of node n_i assigned with channel m_i . The problem can be formulated as follows:

$$\begin{aligned}
 & \text{maximize } f_{s,m_0}, \\
 & \text{subject to: } f_{n_i,m_i} \leq c_{n_i,m_i} \\
 & \forall n_i \in N, f_{n_i,m_{i-1}}^{in} = f_{n_i,m_i} \\
 & SINR_{n_i,m_i} = \frac{S_{n_i}}{I_0 + N_0} > \beta,
 \end{aligned}$$

where I_0 is the interference caused by other nodes, not on the chosen route, working on the same channel with n_i . I_0 is a constant for a given time. f_{s,m_0} is the transmission rate of source node s on channel m_0 , c_{n_i,m_i} is the maximal transmission rate of node n_i on channel m_i , $f_{n_i,m_{i-1}}^{in}$ is the transmission rate coming into n_i and f_{n_i,m_i} is the output transmission rate. Given $SINR_{n_i,m_i}$, $c_{n_i,m_i} = W \log_2(1 + SINR_{n_i,m_i})$. This becomes a linear programming problem and can be solved in polynomial time[64].

We now consider the accumulative interference under the SINR model. To a certain node n_i on the chosen route, the interference caused by other nodes working on the same channel would be taken into consideration. Now the problem can be defined as:

$$\begin{aligned} & \text{maximize } f_{s,m_0}, \\ & \text{subject to: } f_{n_i,m_i} \leq c_{n_i,m_i} \\ & \forall n_i \in N, f_{n_i,m_{i-1}}^{in} = f_{n_i,m_i} \\ & SINR_{n_i,m_i} = \frac{S_{n_i}}{\sum_{j \neq i} a_{m_i,n_j} I_{n_j,n_i} + I_0 + N_0} > \beta, \end{aligned}$$

where a_{m_i,n_j} equals 1 when n_i is using m_i , $\forall n_j \in \{\text{nodes on the chosen route}\}$. Otherwise, it equals 0. Therefore, it considers the choices of nodes on the chosen route.

Theorem 4.7. *The single channel assignment problem under the complex SINR model is NP-hard.*

Proof. Given $SINR_{n_i,m_i} > \beta$, we have:

$$\frac{S_{n_i}}{\sum_{j \neq i} a_{m_i,n_j} I_{n_j,n_i} + N_0} > \beta$$

$$\sum_{j \neq i} a_{m_i, n_j} I_{n_j, n_i} < \frac{S_{n_i}}{\beta} - N_0.$$

Now we can use the reduction from the known NP-hard problem. The General Precedence Constrained SM-RCPS in [65] is NP-hard and can be reduced here. The S_i in SM-RCPS can be represented by the maximal interference among i nodes that are currently assigned channels. Then, the $l_{i,j}$ in SM-RCPS is the increased interference caused by $j - i$ new assigned nodes. \square

In the next part, we make use of the characteristics of DSR and propose an efficient approach.

Single Route Algorithm. There are two main phases in DSR: route discovery and route maintenance. During the route maintenance phase, the destination node would send back a route reply message. Each node on the chosen route can *piggyback* some information to that message. Therefore, the source node can make use of the piggybacked information and assign channels for each node on the route. In general, there are mainly three phases in our approach:

- The source node sends the route request (RREQ) packet through the common channel. After the destination node receives RREQ, it chooses one single route.
- The destination node sends back a route reply (RREP) packet. Each node on the path would piggyback its own estimated preference of each channel to the RREP and forward it to the next-hop node, which is nearer to the source node, on the chosen route through the common channel.
- The source node receives the RREP, including each node's information on the route. It estimates the SINR of each node and calculates the transmission rate based on the estimated SINR. It assigns a channel to each node. The source node distributes the channel assignment result through the common channel.

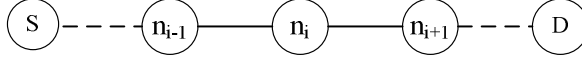


Figure 4.10: An example of single route.

For example, in Fig. 4.10, after destination node D chooses one route, it would send back a RREP along this route. Node n_i would piggyback its preference of each channel and forward to node n_{i-1} . Next, we introduce our approach from two aspects:

- Piggyback: the detailed information of each node piggybacked by RREP;
- Assignment: the channel assignment method after the source node receives the piggybacked information.

The piggybacked information is about each node's preference to channels. Definition 4.8 defines the estimated preference of each node to a certain channel. We denote such preference through a probabilistic way.

Definition 4.8. The preference of node n_i on channel m_i is the probability of n_i choosing m_i as its preferred channel to transmit, denoted by p_{n_i, m_i} :

$$p_{n_i, m_i} = \frac{W \log_2(1 + SINR_{n_i, m_i}^0)}{\sum_{j=1}^M W \log_2(1 + SINR_{n_i, m_j}^0)},$$

where:

$$SINR_{n_i, m_j}^0 = \frac{S_{n_i}}{\sum_{n_k \notin N} a_{m_j, n_k} I_{n_k, n_i} + N_0}.$$

$SINR_{n, m}^0$ is the current minimum SINR in node n 's operation area using channel m . The interference is caused by other nodes, $\{n_k\}$, in the network besides nodes on the chosen route. Thus, $W \log_2(1 + SINR_{n, m}^0)$ is the minimum transmission rate in n 's operation area using m . Here, if a channel's availability dynamically changes or a primary user suddenly appears, the SINR of that channel would become 0. This

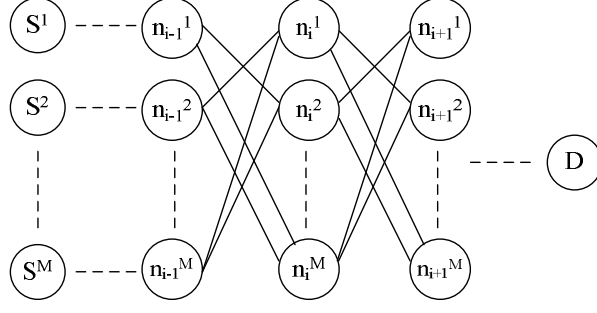


Figure 4.11: Virtual nodes of a single route.

would result in the preference of that channel turning into 0. Therefore, the node would not choose a channel suddenly occupied by a primary user. Besides, we can easily prove that:

$$\sum_{\forall m_i \in M} p_{n_i, m_i} = 1.$$

For example, in Fig. 4.10, after node n_i receives the RREP, it would first add its preference of M channels, $p_{n_i, m_i}, \forall m_i \in M$, to the RREP and then forward to node n_{i-1} .

After the source node receives the RREP, it would assign channels to each node on the route starting from the source node. First, it divides each node into M virtual nodes. Each virtual node occupies a channel. For example, node n_i is divided into $n_i^1, n_i^2, \dots, n_i^M$. Virtual node $n_i^{m_i}$ denotes that n_i occupies channel m_i . Then, the channel assignment on this single route becomes Fig. 4.11. There is no vertical link between virtual nodes of a same node. Definition 4.9 gives the conditions to form horizontal links.

Definition 4.9. If nodes n_{i-1} and n_i are adjacent nodes along the route and n_i is the next hop of n_{i-1} , the horizontal link $(n_i^{m_i}, n_{i-1}^{m_{i-1}}), \forall m_i, m_{i-1} \in M$ only exists if the estimated SINR at any position in the operation area of nodes n_i is above the

threshold β . The estimated $SINR_{n_i, m_i}$ is computed as:

$$\frac{S_{n_i}}{\sum_{j=1}^{i-1} p'_{n_j, m_i} I_{n_j, n_i} + \sum_{k=i+1}^N p_{n_k, m_i} I_{n_k, n_i} + I_0 + N_0},$$

where:

$$p'_{n_j, m_i} = \begin{cases} 0 & m_i \in M' \\ \frac{W \log_2(1 + SINR_{n_j, m_i})}{\sum_{k=1, k \notin M'}^M W \log_2(1 + SINR_{n_j, m_k})} & m_i \notin M' \end{cases};$$

M' is the set of channels that $SINR_{n_j, m'_k} \leq \beta, \forall m'_k \in M'$.

I_{n_j, n_i} is the maximum interference strength a node n_j can produce to any receiver in n_i 's coverage area. S_{n_i} is the minimum received signal strength across n_i 's coverage area. I_0 is the interference caused by other nodes in the network besides nodes on the chosen route. N_0 is the noise level. Nodes $n_j, j \in [1, i - 1]$ are the previous hops of node n_i along the route. Nodes $n_k, k \in [i + 1, N]$ are the following hops of node n_i . p_{n_k, m_i} is the information of node k 's probability of choosing channel m_i , brought by the RREP. p'_{n_j, m_i} is the modified probability of node n_j choosing m_i . This is because after the piggyback phase, the previous nodes have made initial choices about preferred channels based on their estimation and excluded some previous horizontal links. So the p'_{n_j, m_i} is the reevaluated probability for each node on a certain channel based on $SINR_{n_j, m_i}$ instead of $SINR_{n_j, m_i}^0$. It is more precise than the previous estimated value p_{n_j, m_i} .

According to Definition 4.9, the source node forms the links among virtual nodes. It would then compute the route based on the transmission rate on each channel. The expression of the computing transmission rate is $W \log_2(1 + SINR)$, as is stated in the preliminaries.

When assigning channel m_i to a node n_i , previous nodes of n_i have been assigned channels. We use two metrics for channel assignment. One is the maximal interference that the current node can cause to the previous node on a channel. The maximal

Algorithm 7 Channel Assignment of Single Route

- 1: node $n_i =$ source S , $n_{i+1} =$ next hop of n_i
 - 2: $T[m_i]$: an array of size $|M|$, each element is the maximal allowable interference on m_i
 - 3: $L[m_i]$: an array of size $|M|$, each element is the nearest node to next hop on channel m_i
 - 4: **while** $n_{i+1} \neq$ destination D **do**
 - 5: **while** n_i is not assigned any channel **do**
 - 6: $m_i =$ the channel with highest transmission rate
 - 7: **if** $I_{n_i, L[m_i]} < T[m_i]$ **then**
 - 8: assign m_i to n_i
 - 9: **if** $n_i \neq S$ **then**
 - 10: $n_{i-1} = n_i$'s previous hop
 - 11: $n_{i-1} = n_i, n_i = n_{i+1}, n_{i+1} =$ nexthop of n_i
 - 12: Update $T[m_i]$ with minimal $T_{n_j, n_i, m_i}, \forall n_j \in \{\text{previous nodes of } n_i \text{ working on the } m_i\}$
 - 13: $L[m_i] = n_i$
 - 14: Update $SINR_{n_i, m_i} \forall m_i \in M$ based on previous nodes' choices
-

interference caused by current node n_i depends on the distance. Therefore, we save the location information of the node assigned channel m_i , which is the nearest to n_i . Another is the extra maximal allowable interference of previous nodes on a certain channel. The extra maximal allowable interference of node n_j , which is among the previous nodes of n_i on the chosen route and also working on m_i , T_{n_j, n_i, m_i} , is computed based on Definition 4.9 as:

$$\frac{S_{n_j}}{\beta} - \sum_{l=1}^i a_{n_l, m_i} I_{n_l, n_j} - \sum_{k=i+1}^N p_{n_k, m_i} I_{n_k, n_i} - I_0 - N_0$$

The channel assignment process is shown in Algorithm 7. Array $L[m_i]$ keeps the nearest node assigned $m_i, \forall m_i \in M$. Array $T[m_i]$ saves the maximal allowable interference of previous nodes on $m_i, \forall m_i \in M$. After a node is assigned one channel, both the arrays of L and T would be updated. The virtual links are formed and there is only one virtual node linked to the next-hop node. For example, if $n_i^{m_i}$ has a link to the next hop, then n_i is assigned channel m_i .

For example, when assigning channels to n_i in Fig. 4.10, n_i maintains a set of

channels. Its estimated SINR is larger than the threshold β . It would start from the channel with the maximal transmission rate, which can be computed through the SINR. Then, n_i would compute its maximal possible interference $I_{n_i, L[m_i]}$ caused to its nearest node $L[m_i]$, which is assigned m_i . This is also the upperbound of interference that n_i can cause to any previous node. $T[m_i]$ maintains that the lowerbound of the interference can be taken among all previous nodes assigned to m_i . If n_i wants to choose m_i , and $I_{n_i, L[m_i]}$ is lower than the lowerbound of interference that previous nodes on m_i can take, m_i is assigned to n_i . Otherwise, n_i would choose the one with the highest transmission rate among the remaining channels. In the last step, after moving to the next hop, it would first update its SINR estimation on each channel based on previous nodes.

After the source node completes the above process, it would determine the throughput of the chosen route based on the minimum among maximal transmission rate of all the chosen links. We give the correctness and complexity analysis.

First, we would prove that our algorithm above satisfies all of the constraints. f is the throughput after the channel assignment process is completed. $\{c_{n_i, m_i}, n_i \in N\}$ is the set of channel assignment results.

Theorem 4.10. $f_{n_i, m_i} \leq c_{n_i, m_i}, \forall n_i \in N$.

Proof. Since $f_{n_i, m_i} \leq f_{s, m_0} \leq \min\{c_{n_i, m_i}\}, \forall n_i \in N$,

$$f_{n_i, m_i} \leq c_{n_i, m_i}, \forall n_i \in N. \quad \square$$

This is obvious. Next, we need to prove that the SINR of each node after assigning channels is above the threshold.

Theorem 4.11. $SINR_{n_i, m_i} > \beta, \forall n_i \in N$.

Proof. We can prove this by showing that after a certain node n_i is newly assigned channel m_i , $SINR_{n_i, m_i} > \beta$. For nodes before n_i that have already been assigned with channel m_i , also satisfy the SINR constraint.

In the channel assignment phase of Algorithm 7, we would update p' after each node is assigned with a channel. From Definition 4.9, the computation of p' takes the channel assignment results of all previous nodes into consideration. That is, when computing $SINR_{n_i, m_i}$, the interference caused by previous nodes on the single chosen route working on the same channel is considered. Therefore, $SINR_{n_i, m_i} > \beta$.

Then, we need to show that previous nodes assigned with m_i also satisfy the SINR constraint. In Algorithm 7, $I_{n_i, L[m_i]}$ is the maximal interference n_i caused to previous nodes on channel m_i . For any n_j that has been assigned m_i on this route, $I_{n_i, n_j} < I_{n_i, L[m_i]}$. Since $I_{n_i, L[m_i]}$ is less than T_{n_j, n_i, m_i} , I_{n_i, n_j} is less than T_{n_j, n_i, m_i} for any n_j transmitting on m_i , which ensures the interference caused by n_i would not make the SINR of its previous nodes below than β . \square

Now we give the analysis of the complexity of our algorithm, which is shown in Theorem 4.12.

Theorem 4.12. *The worst case complexity of Algorithm 7 is $O(|M||N|^3)$.*

Proof. The complexity of steps 5 and 6 in Algorithm 7 is $O(|M||N|)$. The worst case scenario in step 13 would be to update T for the most times, which happens when the following example situation comes up:

$$m_1 - > n_1, m_2 - > n_2, m_1 - > n_3, m_2 - > n_4, \dots$$

The total update times would be:

$$1 + 1 + 2 + 2 + \dots + \frac{|N|}{2} + \frac{|N|}{2} = O(|N|^2)$$

Therefore, the worst case complexity is $O(|M||N|^3)$. \square

If we find the optimal result by searching all of the different choices, the complexity would be $O(|M|^{|N|})$. Our algorithm is significantly more efficient than the complexity

view when $|M| > |N| > 4$.

We can also have a analysis about the information overhead and computation complexity of the source node. For each node along the route, the information piggy-backed is its preference on each available channel. This is a $|M| \times 1$ metric, which is relatively small. Then, the total information given to the source node is a $|M| \times |N|$ metric. From the above analysis, the computation complexity is $O(|M||N|^3)$. Therefore, the information overhead and computation complexity of the source node is not significant.

4.2.4 Multi-Route Channel Assignment

We first introduce our channel allocation model based on multi-route DSR [66]. It is an extension to the single route and each node chooses its neighbors to help transmit. We would introduce how to convert the multi-route assignment problem to the single route assignment problem above. Then, we give the complexity analysis.

Multi-Route Formation. Compared to the single route, the multi route scheme has two kinds of nodes: (1) base nodes on the main route; (2) alternative nodes on the alternative route. After the destination node chooses a single route, nodes on this chosen route all become base nodes. Each base node chooses its one-hop neighbors to be alternative nodes to help with transmitting. In this way, the throughput is improved and the delay is reduced. We give the specific definition of alternative nodes in our model.

Definition 4.13. For node n_i , if there exists a one-hop neighbor which is not included as the alternative node of n_i 's previous nodes, then this node is n_i 's alternative node, denoted as a_j^i , $j \in [1, \text{number of } n_i\text{'s alternative nodes}]$.

This definition avoids that a node becomes two base nodes' alternative node at the same time. We call this a_j^i being "charged" by node n_i . For example, in Fig.

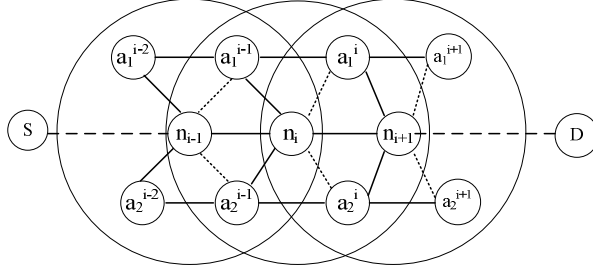


Figure 4.12: Multi-route with alternative nodes.

4.12, although nodes $a_1^{i-1}, a_2^{i-1}, a_1^i, a_2^i$ are all n_i 's one-hop neighbors, only a_1^i, a_2^i are n_i 's alternative nodes. This is because n_{i-1} is the previous node of n_i and a_1^{i-1}, a_2^{i-1} are already charged by n_{i-1} .

The main process is similar to the single route model. The first phase is still the source node sending a RREQ request to reach the destination node. After the destination node chooses a route, it would send back a RREP. Each node on the chosen route now is the base node. When it receives a RREP, it needs to piggyback not only its own probability to choose a certain channel, but also another two aspects of information: (1) its alternative nodes; (2) the probability of alternative nodes choosing a certain channel. From Definition 4.13, two adjacent base nodes have no overlap of alternative nodes. Each alternative node would send its probability of choosing a certain channel to its charging base node. Then, the base node would add all of the information to the RREP and forward it to its next hop on the main route. In Fig 4.12, n_i would add n_i, a_1^i and a_2^i 's probability of choosing each channel to the RREP. Now the channel assignment problem is to assign channels for both base nodes and alternative nodes.

Conversion. The multi-route channel assignment problem can be solved by converting to the single route channel assignment problem with the help of the multistage graph.

As stated in the above part, each base node on the main route charges a set of

alternative nodes. We now define the concept of *stage* in our multi-route problem in Definition 4.14.

Definition 4.14. A stage consists of a base node n_i and the set of alternative nodes $\{a_j^i\}$ charged by n_i .

The links are either within a single stage or within adjacent stages whose base nodes are adjacent in the main route. Each stage is regarded as a node in single route model. Then, the multi-route is converted to a single route. For example, nodes n_i , a_1^i and a_2^i are treated as a single node in Fig. 4.12.

We now need to make modifications to the SINR estimation in Definition 4.9. For each base node, the previous nodes that have influences on its SINR contain both the previous main nodes and previous alternative nodes. The same applies for the SINR calculation of alternative nodes. Previous main nodes and alternative nodes also have influence to its channel choosing probability. Thus, when computing the estimated SINR, we should actually include all of the nodes - base nodes and alternative nodes - in previous stages. Now we define an order to calculate each node's estimated SINR and its probability to choose a certain channel. Starting from the source node, the stage in which the base node is nearer to the source node is computed prior to other stages. During a single stage, the base node would be computed first. It would assign each alternative node an ID, shown in Fig. 4.12. The alternative node with the higher ID number would be computed before other alternative nodes. In Fig. 4.12, nodes n_i , a_1^i and a_2^i would be computed before n_{i+1} , a_1^{i+1} and a_2^{i+1} . In addition, n_i would be computed before a_1^i and a_2^i . a_2^i would be computed before a_1^i due to a_2^i 's ID 2 is higher than a_1^i 's ID 1. Therefore, for a base node, when using Definition 4.9, the p'_{n_j, m_i} should include all of the nodes in the previous sets. For an alternative node, the p'_{n_j, m_i} would include two aspects: (1) all of the nodes in the previous stages; (2) all of the nodes in the same set computed before it.

Locking/Unlocking Scheme. In order to avoid overhearing issues among two

Algorithm 8 Channel Assignment Among Alternative Nodes

- 1: base node $n_i =$ source S , $n_{i+1} =$ next hop of n_i along the main route
 - 2: $T[m_i]$: an array of size $|M|$, each element is the maximal allowable interference on m_i
 - 3: $L[m_i]$: an array of size $|M|$, each element is the nearest node to next hop on channel m_i
 - 4: Lock all the stages
 - 5: **while** $n_{i+1} \neq$ destination D **do**
 - 6: Unlock the set charged by n_i , lock previous stage
 - 7: $A_i =$ set of n_i 's alternative nodes
 - 8: **while** $\exists a_i^j \in A_i$ having no channel assigned **do**
 - 9: $a_i^k = n_i$'s alternative node having no channel assigned with the highest ID
 - 10: Update p' and $SINR_{a_i^k, m_i}, \forall m_i \in M$
 - 11: Choose the channel m_i with highest transmission rate for a_i^k
 - 12: **if** $I_{a_i^k, L[m_i]} < T[m_i]$ **then**
 - 13: assign m_i to a_i^k
 - 14: Update $T[m_i]$ and $L[m_i]$
 - 15: $n = n'$, $n' =$ next hop of n along the main route
-

adjacent stages, we apply a locking/unlocking scheme when assigning channels among different stages.

The source node would assign channels first along the main route, using Algorithm 7. After that, it would assign channels for each alternative node using Algorithm 8. The constraints in Algorithm 8 of $T[m_i]$ and $L[m_i]$ have the same meaning as in Algorithm 7. The update of $L[m_i]$ needs to calculate the minimum distance with the next alternative node to choose which node to store. Each time it would unlock one stage to assign channels and keep other stages locked; nodes in the locked stages would not transmit data so that the unlocked stage would not be interfered with when computing the SINR. After finishing one stage, it would move to the next stage which is charged by the next base node along the main route.

The complexity of our multi-route channel assignment approach is $O(|M|(|N| \times |N_a|)^3)$, where N_a is the alternative node set of a stage with the most alternative nodes among all of the stages. That is, $|N_a|$ is the upperbound of the number of alternative nodes in a single stage. It is obvious that the complexity would be very

Table 4.5: Simulation settings for channel assignment under DSR.

Number of nodes	[100, 300]
operation range of each node	500
Number of channels	[100, 300]
TX power	23 dBm
Noise power	-98 dBm
SINR threshold	10 dB

large when the number of alternative nodes is large in each set.

4.2.5 Simulations

In this part, we perform the simulations for the single route model, multi-route model and the extension of applying virtual backbones. Also, we implement an optimal algorithm, which gets the optimal result via an exhaustive search.

We randomly distribute nodes in a $2,000 \times 2,000$ unit square. Some of the nodes are busy at a certain channel and some of the nodes are idle. We randomly choose a source and a destination. Then, we generate the route along which the channel assignment is conducted. The settings of our simulation parameters is shown in Table 4.5.

The two parameters, number of nodes and number of channels, are tuneable. To compare our algorithm with the optimal algorithm, we change one of the two parameters and compare the algorithms using the metric:

$$U = \frac{f_{s,m_0}}{f_{s,m_0}^o},$$

where f_{s,m_0} is the transmission rate of our algorithms at the source node, and f_{s,m_0}^o is the transmission rate of the optimal algorithm at the source node. Obviously, the higher U is, the closer our algorithm is to the optimal results.

Single Route. We initiate with 100 nodes and 100 channels. The generated route is shown in Fig. 4.13. Here, the number of nodes on the chosen route is 10. Then,

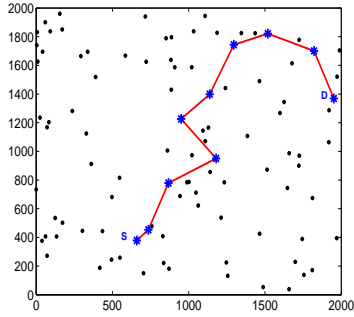


Figure 4.13: Single route.

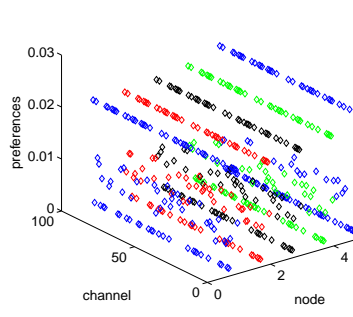


Figure 4.14: Preferences.

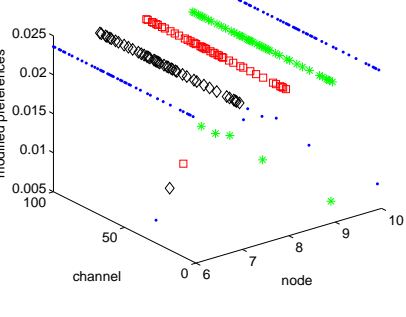


Figure 4.15: Modified preferences.

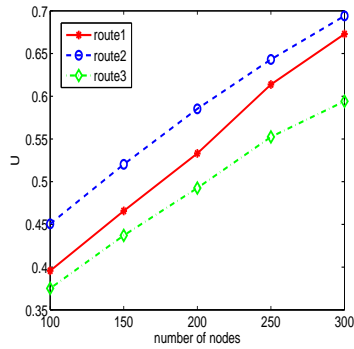


Figure 4.16: U VS nodes.

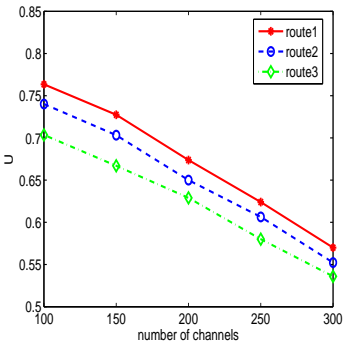


Figure 4.17: U VS Chs.

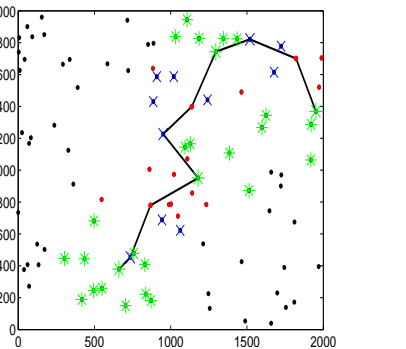


Figure 4.18: Base nodes.

the initial preferences of each node are shown in Fig. 4.14. Due to the consideration of clarity, we only show the five nodes here. From Fig. 4.14, each node can exclude some channels easily (initial preferences = 0).

Based on these settings, we perform our Algorithm 7 from source node to destination node. At a certain point, the preferences of nodes that have not been assigned channels are shown in Fig. 4.15. Here, we choose the 6th node from the source on the chosen route. The previous node has made choices. Their choices modify the preferences of later nodes. The modified preferences are based on a more precise SINR estimation. We only show the preferences greater than 0, which are much less than Fig. 4.14. Therefore, the complexity of assigning channels to later nodes is reduced.

Then, we change the two parameters: number of nodes and number of channels.

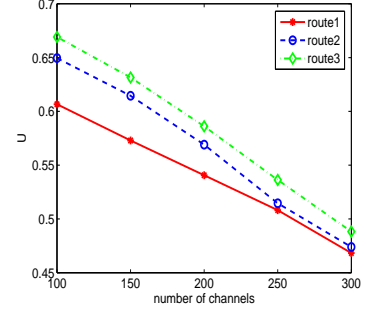
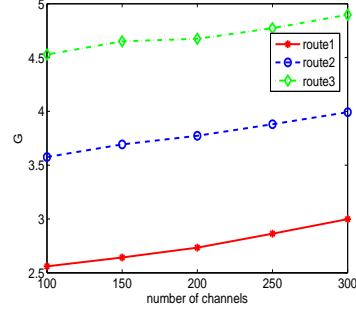
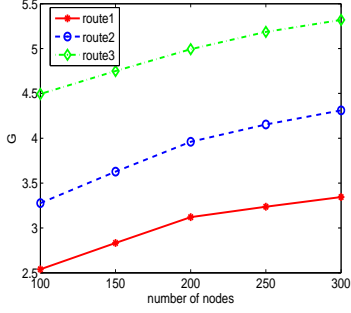


Figure 4.19: G VS nodes. Figure 4.20: G VS Chs. Figure 4.21: U VS nodes.

Each time we generate a new topology and three new routes, we compute the metric U on each route by applying our single route algorithm and the optimal algorithm. Fig. 4.16 shows the results after varying the number of nodes from 100 to 300. When the number of nodes is increased, there would be more interferences in the network; our algorithm is closer to the optimal result. In Fig. 4.17, the number of channels is changed from 100 to 300 and the number of nodes is kept at 200. From the two figures, our algorithm achieves almost 60% of the optimal results.

Multi-route. We use the same setting as the single route. First, we identify the alternative nodes along the main route in Fig. 4.13. The results are shown in Fig. 4.18. We use three colors to distinguish every three adjacent base nodes on the chosen route. Each node's alternative nodes are in the same color. From the figure, two adjacent nodes on the main route have no overlap of their alternative nodes.

Then, we analyze the results after assigning channels for both base nodes and alternative nodes. First, we compare the throughput between the multi-route model and the single route model along three generated routes. The metric used for comparison is as follows:

$$G = \frac{f_{s,m_0}^{mul}}{f_{s,m_0}^{sin}},$$

where f_{s,m_0}^{mul} is the transmission rate at the source node under the multi-route model and f_{s,m_0}^{sin} is under the single route model. We change the two parameters: the

number of nodes from 100 to 300 and the number of channels from 100 to 300. When the number of nodes is changed, the number of available nodes is also changed proportionally. Fig. 4.19 and Fig. 4.20 are the results of three different routes. It is obvious that the throughput of the route is increased under the multi-route model compared to single route model. Moreover, we conduct the optimal algorithm which searches exhaustively and finds the optimal assignment for both base nodes and alternative nodes. We compute the metric U with $f_{s,m_0} = f_{s,m_0}^{mul}$. We change the number of channels from 100 to 300. The values of U are shown in Fig. 4.21. Our algorithm achieves almost 55% of the optimal results.

4.2.6 Conclusion

In this section, we consider the channel assignment problem under dynamic source routing in cognitive radio networks. We make use of piggybacked information to collect information of each node on the chosen route. Also, we propose a mechanism to estimate the SINR, which is used to determine the probability of each node choosing a certain channel. Two models are presented: single route model and multi-route model. We show how to convert the multi-route model into the single route model. Moreover, we propose a locking/unlocking scheme for channel assignment in the multi-route model. Specific simulations are conducted to show the performance of our algorithm. Results show that our algorithms achieve almost 60% of the optimal algorithm in terms of transmission rate.

4.3 Chapter Conclusion

In this chapter, we consider the channel assignment problem in CRNs, which is different from traditional networks due to the high dynamics of channel availability and the low time-consuming requirement. We first consider the conflict-free channel assignment by giving three algorithms. Then, given a specific routing algorithm, we

study the channel assignment for both single route and multi-route scenarios.

We design a fast convergent-localized protocol that assigns conflict-free channels. We first present two basic localized algorithms and an advanced localized algorithm to solve the channel assignment problem. Specifically, we propose a method to partition the given network into “stars” (which resemble 2-level trees) where a localized match between links and channels is feasible. Based on the proposed framework, we further develop three conflict resolution strategies to make our scheme versatily adaptable to different network conditions. We also propose an extension, which can trim nonessential links to further enhance performance.

Under the DSR model, we make use of *piggyback* information in the RREP packet, define the probability of each node choosing a certain channel and use a realistic physical model to estimate the SINR. We study the single route channel assignment problem under our model, which is proven NP-hard, and give the complexity analysis of our proposed approach. Then, we extend to solve a multi-route channel assignment by converting it to the single route channel assignment problem.

In the next chapter, we will present our work on the routing issues in CRNs. Routing in CRNs is different from traditional wireless networks. Based on the works on channel assignment, our proposed routing protocols are spectrum aware with high reliability.

CHAPTER 5

SPECTRUM AWARE ROUTING IN CRNS

Since nodes in CRNs need to quit immediately when PUs become active and occupy a channel, the links are highly unstable for data transmission. The dynamics of channel availabilities result in the difficulty of routing in CRNs, and in carrying out end-to-end data transmission. First, the routing protocol should be able to select the initial route while considering the influences of PU activities in future. In addition, the route reliability and route recovery cannot be neglected considering the inevitable route broken issues during transmission. In this chapter, we propose our works on designing efficient routing protocols in CRNs.

5.1 Routing Protocol with Boundary Nodes

There have been many works on routing in CRNs [5]. Since the dynamic channel availabilities affect the delay and reliability of each route, the route selection algorithm needs to consider the channel availability situation of each optional route. Most of the existing routing protocols rely on the piggybacked channel information, and build their metrics regarding multiple parameters, e.g., channel availability and route quality. Then the route selection is usually based on these metrics. However, there are two main problems with these protocols. First, the overhead of the piggybacked information is usually too large, which makes it impractical when considering the energy and interference. Second, the piggybacked channel availabilities cannot convey the instant channel situation because, at the time the data is being transmitted, the channel availabilities are possibly different. Therefore, a better protocol should be

able to take both the overhead and channel availability dynamics into consideration.

We consider the routing problem in a novel way, and make use of the directional antennas to help route selection. There have been many works done using directional antennas to benefit the data transmission in traditional wireless ad hoc networks[67]. There are two benefits to applying directional antennas. One is the reduction of radio interference. Thus, in CRNs, it increases the spacial reuse opportunities among PUs and SUs, and also for SUs themselves. Another benefit of directional antennas lies in the determination of if a node is located at the boundary of a PU area. The different directional antennas on each node do not need to be globally aligned, which is similar to the directional antenna model in [68].

We assume that each node is equipped with a directional antenna, which is a reasonable assumption, considering the directional functions in many mobile devices today. We define the boundary node, and each node is able to decide whether it is, itself, a boundary node or not. We use the USRP/Gnuadio testbed to show the difference among sensing results in different sectors of the directional antenna on a boundary node. We make use of the piggybacked information by boundary nodes. The source node makes use of the information returned by boundary nodes along each possible route, measures the channel availability and stability of each route, and chooses the best one to reach the destination node through our algorithm. During the process of route selections by the source node, the boundary nodes' information can be used as a "traffic blocker", which "blocks" traffic from entering too many PU areas.

5.1.1 Preliminaries

The delay of a routing path in CRNs is different from that in traditional wireless networks, due to the dynamic channel availability on each node. The main factors that influence the delay are the following:

- Medium access delay, which is the delay when a node accesses a given channel;
- Queueing delay, which is related to the output transmission rate of a node on a given channel;
- Handoff delay, which happens when the current channel is occupied by PUs, and the nodes need to switch to another channel;
- Rerouting delay, which is when the current link is unable to meet the transmission requirements, and the sender needs to seek another route to reach the receiver.

Therefore, if a routing path consists of many unstable links that need to be frequently handed off or rerouted, the overall delay would increase because of the increment in medium access delay, handoff delay, and rerouting delay. If the sender of a given link can only use low power to send, due to the interference requirements of PUs, this would result in low transmission rate, and the queueing delay would increase.

5.1.2 Problem Formulation

We consider a CRN with the node set, $\{a, b, c, \dots\}$. Each node is equipped with directional antennas, which divides the omnidirectional transmission range of each node into a number of sectors. We assume that each node is static. The total available channel set is denoted as M , which is licensed to a set of PUs, whose activities are unknown. During the data transmission, each node selects one sector to send the data. We assume that there is a CCC for nodes to coordinate. Our model can also be extended to environments without a CCC, using the channel rendezvous approaches in [69]. We assume the existence of a CCC here for simplicity.

There are several constraints that need to be satisfied for successful data transmission, regardless of which sector is adopted by each user. When node a sends data

to node b , they must tune in to the same channel, $m \in M$. Suppose the power used by node a is P_a . The minimum SINR requirement at b is α_b . Therefore, we have

$$\frac{P_a g_{ab}}{N_0 + I} > \alpha_b. \quad (5.1-1)$$

Moreover, suppose that the nearby primary user pairs, $PU - TX$ and $PU - RX$, are working on the same channel m . Then, we have

$$\frac{P_p g_{pp}}{N_0 + I + g_{ap} P_a} > \alpha_p, \quad (5.1-2)$$

where P_p is the power adopted by $PU - TX$, g_{pp} is the power gain from $PU - TX$ to $PU - RX$, g_{ap} is the power gain from node a to $PU - RX$, and α_p denotes the desired SINR requirement of $PU - RX$. The data transmission from a to b is successful only if the above two constraints are satisfied.

Since the position of any $PU - TX$ is unknown to SUs, to make sure the PU sessions are not interfered with, we strengthen Constraint (5.1-2) as for any point within $PU - TX$'s area, instead of only $PU - RX$, where the SINR value is above α_p ; P_a must satisfy Constraint (5.1-2), no matter which sector a uses. Therefore, when a and $PU - TX$ are working on the same channel, we have the following three situations regarding the constraint of P_a , based on the distance between $PU - TX$ to node a :

$$P_a = \begin{cases} 0 & \frac{P_p g_{pa}}{N_0 + I} > \alpha_p, \\ P'_a & \frac{P_p g_{pp'}}{N_0 + I} = \alpha_p, g_{ap'} P'_a \rightarrow 0, \\ P_{max} & \frac{P_p g_{pp'}}{N_0 + I} = \alpha_p, g_{ap'} P_{max} = 0. \end{cases} \quad (5.1-3)$$

The first case is that when the SINR value of $PU - TX$ at a 's location is above α_p , a cannot use the channel of $PU - TX$. The second case is that a can use the channel of $PU - TX$, but the interference caused by a cannot make any point that has a

SINR greater than or equal to α_p as being less than α_p . p' is a boundary point of $PU - TX$'s transmission area, where the SINR is equal to α_p . The third case is that a is far from $PU - TX$'s transmission area. Node a can transmit at the maximal power P_{max} without causing any significant interference to any point within $PU - TX$'s transmission area.

Suppose there are session requests in the CRN. For a source node S , the objective of our model is to find the route with the minimal delay while ensuring the reliability, as to reach the destination node, D . Based on the delay discussion in previous subsection, the channel availabilities on each link play an important factor in determining the overall delay. Since the channel availabilities on each link are dynamic, it is impractical to find the optimal solution. Even if a route provides the minimal delay at a given time, it is unable to ensure the minimal delay during the entire session.

Without loss of generality, we assume the data transmission delay for a single link is a constant, if there is no PU, based on the constraints in Eq. (5.1-3). Our model can be easily extended to variable data transmission delay, (e.g., with power management), which will be shown later. Therefore, the total delay of each route is proportional to the number of hops plus the handoff or rerouting delay, caused by active PUs. An intuitive idea is to have nodes on all potential routes piggyback their channel availability information, and let the source node collect all the information to make decisions about route selections. This solution has two defects: 1) when the distance between the source node and the destination node is very large, the information of all nodes could be huge and burden the CCC; 2) the channel availability information of each node is dynamic in CRNs. It means that the channel availability could change during data transmission, which would cause longer delay and even break the routes.

We provide a protocol for routing which considers both delay and reliability, with the help of directional antennas. In our model, to select a route, there are four

factors that need to be taken into consideration: the nodes on the route, the sector adopted by each node to transmit, the channel used on that sector, and the power allowed on that channel. Considering that the interference constraints of PUs and SUs are both dynamic, it is impractical to find the optimal solution. We propose an effective routing protocol, which makes use of the directional antennas, and efficiently reduces the overhead during the exchange of control messages on CCC. Our model focuses on piggyback and route selection phases: after the route is selected, the channel management and channel selection for each single link is out of scope. Works in [70, 71] can be easily applied for data transmission on the selected route under different network environments.

5.1.3 Routing Protocol with Directional Antenna

Since each node in our model is equipped with a directional antenna, for a node a , we use the 2-tuple (IN_a, OUT_a) where IN_a denotes the sector ID that a packet is received by a , and OUT_a denotes the sector ID that the packet is sent out by a .

Similar to the source routing in traditional ad hoc networks[59], in our model, the source node first needs to find the route to reach the destination node using the following process:

- The source node, S sends the route request (RREQ) packet through the CCC from all sectors. The RREQ contains the ID of the destination node D , denoted as $\langle S, D \rangle$. Also, for every sector to which the RREQ is sent out, it also contains the OUT_S . Since it is the source node, the IN_S is empty.
- For any node a that receives the RREQ, it would add its own node ID and broadcast the request through all of its sectors. Moreover, it would also add (IN_a, OUT_a) to the RREQ. Obviously, IN_a is the same for all sectors, from which the RREQ is received. OUT_a differs among the RREQs from different sectors.

- The above two processes continue until the destination node is reached. Then the RREQ will contain all the node IDs from S to D , denoted as $\langle S, \dots, a, \dots, D \rangle$. The RREQ also contains the IN and OUT sector IDs of each node on the path, for example, the (IN_a, OUT_a) of node a . In addition, for destination node D , it only contains IN_D , since the OUT_D is empty.

After the destination node D is reached, it will reply with the route reply (RREP) packet over CCC, along with the route information (node IDs and sector numbers) in the RREQ packet, to the source node. The underlying MAC protocol can make use of works in [67], since the RREQ and RREP messages are sent through CCC from all sectors, which convert the MAC problem very similar to that in the traditional wireless networks. The relay node selections and avoiding of *RREQ* cycles can be performed by applying the approach in [72]. Since, in most cases, there are several routes from S to D , the source node S needs to select one of them. It is intuitive to consider adding the channel availability situation of the corresponding sector on each node to the RREP message along the route, and piggybacking it to S . However, this is impractical. Since the channel availabilities on each sector of each node are dynamic, the channel availability of each node can differ between the piggyback phase and the data transmission phase. Also, it would cause lots of overhead to return the channel availability of every node on the route. In our model, we make use of the directional antennas, and propose an efficient route selection scheme.

Boundary Node. We first give the definition of *boundary node* under our model. There are many existing boundary detection algorithms [73]. Our definition here mainly describes what the boundary we refer to in CRNs.

Definition 5.1. Node a is a boundary node regarding the $PU - TX$ on channel m if the variance, $V_a(m)$, of the sensing results in all sectors of node a is above a threshold, ν . We use $B_a(m) = 1$ to denote that a is the boundary node of $PU - TX$

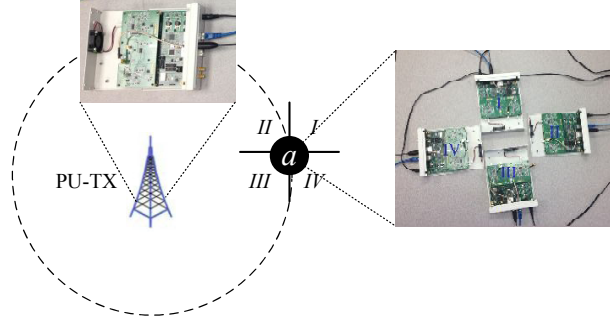


Figure 5.1: Testbed for showing the characteristic of a boundary node.

that occupies channel m . Then

$$B_a(m) = \begin{cases} 0 & V_a(m) \leq \nu, \\ 1 & V_a(m) > \nu. \end{cases} \quad (5.1-4)$$

From the definition, we can see that the boundary nodes are a region of nodes, rather than a line of nodes. The variance here refers to the variances of sensing results among different sectors, as seen in the following experiment. We use the USRP/Gnuradio testbed to show the difference of the received SINR at different sectors of a boundary node. As shown in Fig. 5.1, to simulate a SU with a four-directional antenna, we use four USRP N200s, and each of them denotes one sector. Another USRP N200 is used to simulate a $PU - TX$. We use narrowband communication here. The PU sends on the channel with central frequency 1.3005GHz, and the other 4 USRP N200s receive at the same channel. The received SINRs at the USRP N200s located at sectors I and II are shown in Figs. 5.2 and 5.3. We use the waterfall plot to show the SINR values over time. The approximate SINR at sector I is about -50 dB, while the value at sector II is about -87 dB. The differences of SINR values over time at different sectors of a boundary node are very obvious.

Many merits of boundary nodes have been studied in traditional wireless networks[74, 75]. In addition, boundary nodes in CRNs can facilitate the routing path selection.

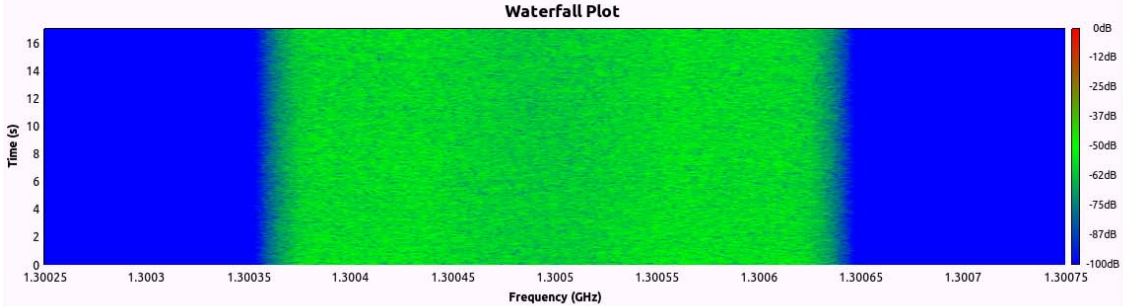


Figure 5.2: Receiving results at sector *I*.

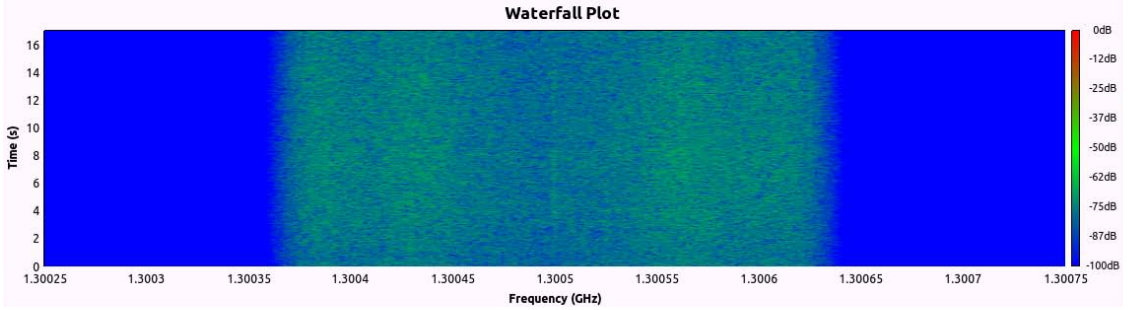


Figure 5.3: Receiving results at sector *II*.

They can help to differentiate the routes that go into or avoid the $PU - TX$'s area. For example, in Fig. 5.4, suppose that the two $PU - TX$ s occupy channel m_1 and m_2 , and are randomly active. Route R that goes out from sector *III* of node c is different from route R' that goes from sector *IV*. Intuitively, when channel m_1 is unavailable, route R is better than route R' because the following links of c on route R' are more likely to be broken, which is unreliable and would cause more delay.

Threshold-Based Piggyback Scheme. Having the boundary node definition, each node can identify if it is, itself, a boundary node of a certain primary user during the spectrum sensing phase. Our protocol will make use of boundary nodes during the piggyback phase.

As stated at the beginning of this section, when node a receives the RREQ packet, it will add both its ID and the 2-tuple (IN_a, OUT_a) to the RREQ. However, if a is a boundary node of the PU occupying channel m , and the active probability of that PU is above a predefined threshold γ , it will add the 4-tuple $(IN_a, OUT_a, m, \mu_a(m))$

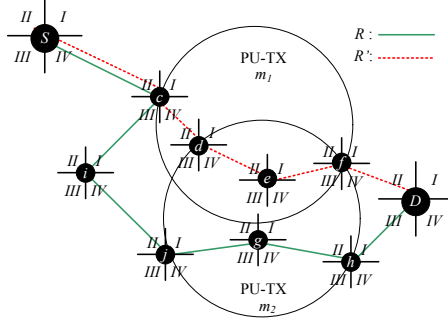


Figure 5.4: Two possible routes from S to D .

to the RREQ, where $\mu_a(m)$ equals 1 or -1, indicating the entrance to, or exiting of, the PU area of channel m . More specifically, we provide the following three cases for node a to decide which information it will add to the current RREQ packet:

- If $\exists m \in M$ that satisfies $B_a(m) = 1$ & $PB_a(m) > \gamma$, and m is unavailable on the sector number OUT_a from which the RREQ is sent out, rather than a 's ID and its 2-tuple (IN_a, OUT_a) , a would add the 4-tuple $(IN_a, OUT_a, m, \mu_a(m))$ to the RREQ. Here, m is the channel that is unavailable in sector OUT_a , and $\mu_a(m) = 1$, which indicates the entrance to the PU area.
- If $\exists m \in M$ that satisfies $B_a(m) = 1$ & $PB_a(m) > \gamma$, and m is unavailable on the sector number IN_a from which the RREQ is received, a would add its ID and the 4-tuple $(IN_a, OUT_a, m, \mu_a(m))$ to the RREQ, where m is the channel that is unavailable in sector IN_a , and $\mu_a(m) = -1$, which indicates an exit from the PU area.
- Otherwise, a would only add its ID and the 2-tuple (IN_a, OUT_a) to the RREQ.

The first case presents the situation in which the route enters the PU area occupying m , reported by the boundary node a . The second case represents the situation in which the route leaves the PU area occupying m , reported by the boundary node a . In both cases, the PU occupying m does not have to be active at the time when RREQ is transmitted, as long as they are previously measured by the boundary nodes

and their active probability measured by a is above a predefined threshold γ . The third case is for nodes that are not boundary nodes, or nodes that are boundary nodes but the active probability of PU is below the threshold γ , regardless of if they are inside or outside the PU areas.

The reason that the active probability of PU during T has to be above the predefined threshold γ is because different PUs have different active levels. For example, some PUs are active much less frequently than other PUs. It is possible that entering these PU areas could achieve a better performance than choosing other routes, which do not go through those PU areas but take longer hop distances to reach the destination. The route selection algorithm is discussed in detail in the following parts.

For example, in Fig. 5.4, the two $PU - TX$ s occupy channels m_1 and m_2 . There are two optional routes, R and R' , from source S to destination D . Node j in Fig. 5.4 satisfies the first case, where $B_j(m_2) = 1$, and m_2 is unavailable on sector I ($OUT_j = I$). j would add its ID and the 4-tuple $(II, I, m_2, 1)$ to the RREQ. Node h in Fig. 5.4 belongs to the second case. Thus, h would add its ID and $(II, I, m_2, -1)$ to the RREQ. Node i in Fig. 5.4 meets the conditions of the third case. Therefore, i only adds its ID and (I, IV) to the RREQ. The burden of CCC is reduced since only boundary nodes are required to return extra information.

After the destination node D is reached, it copies the route information and the added 2 or 4-tuple information by each node in RREQ, and piggybacks to source node S in RREP. Using route R in Fig. 5.4 as an example, the RREP would contain the node IDs, $\langle S, c, i, j, g, h, D \rangle$, on R , and also the 2 or 4-tuple attributes of each node. Among all nodes on route R , j has the 4-tuple $(II, I, m_2, 1)$, and h has the 4-tuple $(II, I, m_2, -1)$. The others have 2-tuple, indicating the IN and OUT sector numbers.

Route Length. After the source node receives the RREP along with piggybacked information, it needs to perform the route selection, since there is usually more than

one route that can reach the destination node.

Due to the dynamics of channel availabilities, it is impractical to estimate the delay of each route and choose the optimal one. To achieve our goal, we provide a heuristic approach to estimate the delay of each route. There are two factors to be considered: the delay of each link along the route, and the length of the route. The transmission rate depends on the power used by the sender. The maximal power that a node can use happens when that channel is free of the corresponding PU area. Intuitively, the route that passes through the lower amount of PU area is more likely to achieve less delay for each link. Instead of adopting the traditional way of calculating the route length, we propose a novel weighted route length calculation, which connects the channel availabilities of each route with the route length.

The calculation of the route length is conducted by the source node, which makes use of the information contained in the piggybacked RREP packet. We use ab to denote a single link from a to b on the optional route. a and b have a 2-tuple or 4-tuple attribute, depending on whether it is a boundary node or not.

We start by defining whether a link is inside or outside a PU area that occupies channel m . The source node treats the nodes that return a 2-tuple as a non-boundary node. For example, as discussed above, if a node a is a boundary node but the active probability of PU is below a threshold, it only returns a 2-tuple. The source node would treat a as a non-boundary node, as well as other real non-boundary nodes.

Definition 5.2. A single link ab is inside the PU area that occupies channel m if any of the three cases are satisfied:

- $B_a(m) = 1, \mu_a(m) = 1$;
- $B_a(m) = 0, B_b(m) = 1, \mu_b(m) = -1$;
- $B_a(m) = 0, B_b(m) = 0$, and $\exists c$, which satisfies $B_c(m) = 1$; c is the boundary node nearest to a among all the boundary nodes before a on the given route,

and it satisfies $\mu_c(m) = 1$.

Otherwise, the link is outside the PU area of m . For a given route, we use $I_{ab}(m) = 1$ to denote that the link ab is inside the PU area of m , and $I_{ab}(m) = 0$ to denote that it is outside the PU area of m .

In the above definition, the first case means that node a is the starting point of entering the PU area occupying m . The second case means that node b is the end point of leaving the PU area occupying m . The third case means that the link is inside the PU area, and neither of the two nodes is the boundary node. $B_a(m) = 0$ is necessary in the second case, which eliminates the case that $B_a(m) = 1$, $\mu_a(m) = -1$, $B_b(m) = 1$, and $\mu_b(m) = -1$, that is, both a and b denote leaving the same area. Links under this case should be considered as being outside the area of channel m . The source node S and the destination node D do not need to be considered, since they are not optional nodes on the route.

Next, we give examples of the three cases, and one special case (Case 4), in which a link is located within multiple PU areas in Fig. 5.5:

- Case 1: Link ab is in the PU area since a is a boundary node and $\mu_a(m) = 1$, which means link ab has entered the PU area of channel m and is within it;
- Case 2: Link ab is in the PU area since node b is a boundary node and $\mu_b(m) = -1$, which means link ab is also in the PU area of channel m ;
- Case 3: Link ab 's closest boundary node for the PU area of channel m is c and $\mu_c(m) = 1$, which means link ab is in the PU area;
- Case 4: For channel m_1 , similar to Case 3, link ab 's closest boundary node is d and $\mu_d(m_1) = 1$; for channel m_2 , link ab 's closest boundary node is c and $\mu_c(m_2) = 1$. Therefore, link ab is within two PU areas of channel m_1 and m_2 .

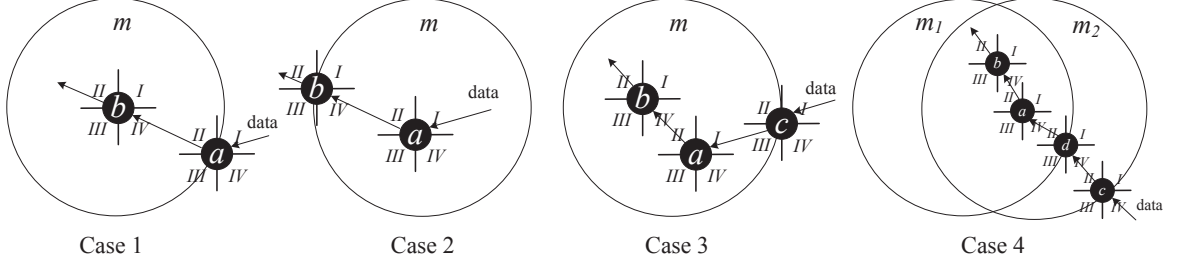


Figure 5.5: Four cases of link ab located in one or multiple PU areas, detected by boundary nodes.

From the above discussions, the source node can tell the number of PU areas that a single link passes through. Based on that, we define the weighted length of a single link, which will be used later to define the weighted length of a route. For a node that is a boundary node of multiple PUs, e.g., two PU areas, then it replies with two 4-tuples to indicate the entering or leaving the two PU areas.

Definition 5.3. For a single link ab on a given route, the weighted length of the single link L_{ab} is calculated as:

$$L_{ab} = \begin{cases} 1 & I_{ab}(m) = 0, \forall m \in M; \\ \frac{|M|}{|M| - \mathcal{C}(m)} & I_{ab}(m) = 1; \end{cases} \quad (5.1-5)$$

where $\mathcal{C}(m)$ counts the number of channels on link ab that satisfy $I_{ab}(m) = 1$, which means that ab is inside the PU area of m , and $|M|$ is the number of total channels in the network.

Therefore, we have the definition of the weighted length of a single route.

Definition 5.4. For a route R , the length of R is:

$$L(R) = \sum_{ab \in R} L(ab), \quad (5.1-6)$$

where ab is any link on the route R .

Algorithm 9 Route selection from route set \mathcal{R} .

Require: \mathcal{R} , the route set;**Ensure:** *Route*, the selected route;

- 1: $Length = \infty$, $Route = null$;
 - 2: **for** $R \in \mathcal{R}$ **do**
 - 3: Calculate $L(R)$ using (5.1-6); {C}calculate the route length of every R
 - 4: **if** $L(R) < Len$ **then**
 - 5: $Route = R$;
 - 6: $Length = L(R)$;
 - 7: **return** $Route$ {R}eturn the route with minimum length.
-

From the definition of the weighted route length, we can see that both the number of hops to reach the destination and the channel availabilities on the route are taken into consideration.

Route Selection Algorithm. For the source node S , after it receives the RREP from the destination node D , it will retrieve the information in the multiple RREP messages, and select one route to reach D . Suppose that the set of routes S can select from is \mathcal{R} . The algorithm for S to select a route from \mathcal{R} is in Algorithm 9. It will choose the route with the minimum weighted length, based on the definition in the previous part.

We use Fig. 5.4 as an example. Suppose there are 3 channels in total, which means $|M| = 3$. The weighted length values of each link on two optional routes, R and R' , are shown in Table 5.1. Since $L(R) = 7$ and $L(R') = 19/2$, R is chosen as the route from S to D . The advantage of R can be seen when both PUs become active; links de and ef on R' only have one same channel available, making them unable to transmit simultaneously due to interference. Also, if another primary user suddenly joins this area, then R' is more likely to have broken links, which would cause more delay.

After the route is selected, each node on the selected route will sense the channels, and choose the one with the maximum transmission rate, based on the constraints.

Performance Analysis. We analyze the reliability increment of one link, on av-

Table 5.1: An example of weighted route length.

R	Sc	ci	ij	jj	gh	hD
7	1	1	1	$\frac{3}{2}$	$\frac{3}{2}$	1

R'	Sc	cd	de	ef	fD
$\frac{19}{2}$	1	$\frac{3}{2}$	3	3	1

erage, by comparing the route selected by our algorithm with the traditional shortest path.

Given a pair of source node S and destination node D , let N denote the node set of the route selected by our algorithm, and N_0 denote the node set of the shortest route selected by a traditional greedy algorithm. Then we have:

$$N = N_0 + \Delta, \quad \Delta > 0, \quad (5.1-7)$$

where Δ is the difference in the number of nodes.

Theorem 5.5. *We use \mathcal{Q} to denote the average number of PU areas that a single link on N is located inside, and \mathcal{Q}_0 for N_0 . Then*

$$\frac{\mathcal{Q}_0}{\mathcal{Q}} > 1 + (\eta - 1) \frac{\Delta}{N}, \quad (5.1-8)$$

where $\mathcal{Q} = |M|/\eta$, $|M|$ is the total number of channels, and $\eta \geq 1$.

Proof. It is obvious that $|M| = \eta\mathcal{Q}$, and $\eta \geq 1$, since $|M|$ is the maximum possible number of channels that a single link can possibly be inside. From the definition in (5.1-6), and Algorithm 9, which ensures that the route consisting of \mathcal{N} has the minimum weighted length:

$$\frac{|M|}{|M| - \mathcal{Q}_0} \times N_0 > \frac{|M|}{|M| - \mathcal{Q}} \times N$$

$$\frac{|M|}{|M| - \mathcal{Q}_0} \times N_0 > \frac{|M|}{|M| - \mathcal{Q}} \times (N_0 + \Delta)$$

$$\frac{\mathcal{Q}_0}{\mathcal{Q}} > \frac{N_0 + \frac{\Delta|M|}{\mathcal{Q}}}{N}$$

Since $|M| = \eta\mathcal{Q}$,

$$\frac{\mathcal{Q}_0}{\mathcal{Q}} > \frac{N_0}{N} + \frac{\Delta\eta}{N}$$

Given $N = N_0 + \Delta$, then

$$\frac{\mathcal{Q}_0}{\mathcal{Q}} > 1 + (\eta - 1)\frac{\Delta}{N}.$$

□

From Theorem 5.5, when the PUs suddenly appear, the larger η will ensure a more reliable performance of Algorithm 9, compared to the traditional shortest path. This is because a larger $\mathcal{Q}_0/\mathcal{Q}$ provides a lower probability that a suddenly appearing PU will break a link on the route between S and D . A more reliable link reduces the delay, since the node of a broken link needs to perform handoff or rerouting to reach the next hop. The selection of each link in our scheme gives a more reliable link, when facing the suddenly active PUs. Therefore, our algorithm can adjust better under the dynamic environment of channel availabilities in CRNs.

5.1.4 Feasibility Improvement

To improve the feasibility of our model, in this section, we propose schemes solving two situations with imperfect information. The first one is the imperfect detection of PU boundaries. The second one is the imperfect detection of boundary links, whose end nodes are all boundary nodes.

Virtual Boundary Node. It is possible that in a sparse network, not all the boundaries of PUs are detected by nodes in CRNs. When boundaries are not detected, it could cause severe impacts to the weighted length calculations of single links and different routes. The route selected by Algorithm 9 would not be the one that passes

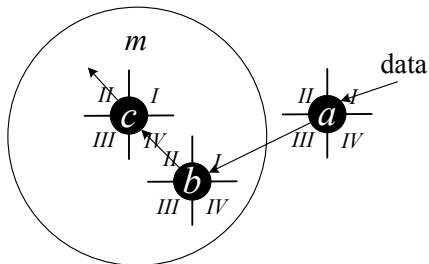


Figure 5.6: An example of missing boundary detection for the PU area of m .

the least number of PU areas. Therefore, we propose the “virtual boundary node” scheme, which takes a limited extra communication cost, to overcome the missing boundary detection problem.

Fig. 5.6 is an example of missing boundary detections. Node a is outside the PU area of channel m , and node b is inside the PU area. Since there is no boundary node of link ab , and ab enters the PU area, the link bc after ab , which is inside the PU area, cannot be detected as being in the PU area of channel m .

To introduce our solution, we first define the virtual boundary node based on the links that pass across the PU boundaries:

Definition 5.6. For a link ab that crosses the PU boundary of channel m , node b is the virtual boundary node, which is the next hop node of a .

For a link ab , if it crosses a PU boundary, then there are two possibilities according to the direction of the link: either ab enters the PU area, or exits from the PU area. Based on the above virtual boundary node definition, if link ab enters the PU area, it means that a is outside the area and b is inside. If ab exits from the PU area, then a is inside the area and b is outside. In both cases, based on Definition 5.6, b is the virtual boundary node. For example, in Fig. 5.6, the virtual boundary node is b .

If node b is a virtual boundary node, although it is not a real boundary node, itself, it would piggyback the 4-tuple information to the source node, which is similar to a boundary node, instead of its 2-tuple information. Therefore, if node b is a virtual boundary node for the PU area of channel m , then we set $B_b(m) = 1$, and

Algorithm 10 Virtual boundary settings for node b on link ab .

Require: M_a, M_b , the available channel sets of a and b ; M , the total channel set;

Ensure: $B_b(m), \forall m \in M$; $\mu_b(m)$, the associate value when $B_b(m) = 1$;

```

1: for  $m \in M$  do
2:   if  $m \in (M_a - M_b)$  then
3:      $B_b(m) = 1, \mu_b(m) = 1$ ;
4:   else if  $m \in (M_b - M_a)$  then
5:      $B_b(m) = 1, \mu_b(m) = -1$ ;
6:   else
7:      $B_b(m) = 0$ ;
8: return  $B_b(m)(\forall m \in M), \mu_b(m)$  if  $B_b(m) = 1$ ;

```

the corresponding value of $\mu_b(m)$ is similar to a real boundary node. Our scheme to overcome the missing boundary detection problem only requires the information exchange of one-hop nodes. Given a link ab , node a sends its available channel set, M_a , to node b . Node b uses its own available channel set M_b and runs Algorithm 10, to decide if it is a virtual boundary node, itself. If it is, what the value of $\mu_b(m)$ is.

From Algorithm 10, if node b is a virtual boundary node, or $B_b(m) = 1$, then the value of $\mu_b(m)$ has the same meaning of a real boundary node. That is, if $\mu_b(m) = 1$, then link ab enters the PU area of channel m . If $\mu_b(m) = -1$, then link ab exits from the PU area of channel m . For example, in Fig. 5.6, since $m \in (M_a - M_b)$, then $\mu_b(m) = 1$, which means link ab enters the PU area. During the piggyback phase, the virtual boundary node will piggyback the 4-tuple information, similar to a real boundary node. For example, node b in Fig. 5.6 would piggyback $(I, II, m, 1)$.

Overall, our scheme indeed causes an extra communication cost to overcome the boundary missing detection problems. However, it only requires the available channel set exchanges among one-hop nodes. A node can decide if it is a virtual boundary node, itself, based on the available channel set of its previous one-hop node on the route. For example, nodes a and c in Fig. 5.6 do not need to exchange information, or know each other's available channel set. Our scheme, based on virtual boundary nodes, can be easily extended from the previous model, since virtual boundary nodes

are treated the same as real boundary nodes and the route selection algorithm is unchanged.

Threshold-based Boundary Link. Given a link ab , it is possible that both end nodes are boundary nodes, and link ab itself is located at the boundary of a PU area. Under this situation, it is impractical to simply count link ab as inside or outside the PU area. Therefore, we propose a heuristic solution, which is threshold-based.

The threshold we use here is based on the distance of a and b . If the distance between a and b is above the threshold, then link ab is counted as a link inside the PU area. Otherwise, ab is treated as outside the PU area. This will affect the piggyback information of a and b , since they are both boundary nodes, as shown in the following two cases:

- If ab is treated as inside the PU area of channel m , node a would piggyback the 4-tuple information with $B_a(m) = 1$ and $\mu_a(m) = 1$, indicating the entering of this PU area. If the next-hop node of b on the route is inside the PU area, b would only return 2-tuple information, as a non-boundary node. Otherwise, b would piggyback the 4-tuple information with $B_b(m) = 1$ and $\mu_b(m) = -1$, indicating an exit from the PU area.
- If ab is treated as outside the PU area of channel m , node a would only return 2-tuple information, as a non-boundary node. The piggybacked information by b depends on the next-hop node of it on the route, which is similar to the previous case.

One thing to notice is that, the threshold we use here can be changed under different bases, e.g., transmission power, or the distance to PUs, according to different requirements.

5.1.5 Extensions

In this subsection, we discuss two promising extensions of our work. One is the error detection from the piggybacked information. The other is the application of different physical layer models.

Error Detection. Our model assumes that the sensing results of different nodes, especially boundary nodes, are always correct. However, in real scenarios, it is possible that some sensing results are incorrect, e.g., the false positive and the false negative errors. The false negative error is that a node's sensing results indicate the presence of a PU while the PU is actually not there. The false positive error is that a node's sensing results indicate that there is no PU on one channel, while in fact, the PU is there. This could cause the boundary nodes, or virtual boundary nodes, to claim the entering or exiting from PU areas incorrectly, which would affect the results of route length calculations and route selections.

Detection of these errors can be performed by the source node. Moreover, it requires the destination node D to piggyback its own sensing results. Since the piggybacked information contains the entering and exiting from PU areas of different channels, the source node can predict the PU areas that it locates at, itself. Then, based on the prediction results, the source node can compare with its own sensing results. If the two results do not match, it means there is error in the sensing results, either by the source node, or by other boundary and virtual boundary nodes on the route. The overview of the source node, S , detecting the errors on one route is:

1. For each channel $m \in M$, the source node S collects the piggybacked information, which includes the sensing results of destination node D , and sums up all the $u_a(m)$, if $B_a(m) = 1, \forall a$ on the route;
2. Source node S maintains two lists, L_{in} and L_{out} , to store the channels whose sums from Step (1) are not 0. If the sum of a channel is greater than 0, the

channel is stored in L_{in} . If the sum is negative, the channel is stored in L_{out} ;

3. S compares the two lists, L_{in} and L_{out} , with its own sensing results. For every channel $m \in L_{in}$, if destination node D 's sensing results show that m is available. Then, there is an error on the sensing results of m , either by S , or boundary nodes, including virtual boundary nodes, on the route. For every channel $m \in L_{out}$, if node S 's sensing results show that m is available, similarly, there is also an error.

For a boundary node a with $B_a(m) = 1$, if $u_a(m) = 1$, it means the route enters a PU area of channel m . If $u_a(m) = -1$, it means the route exits from the PU area. If channel m is in L_{in} , it means that the sum of $u_a(m)$, $\forall a$ on the route with $B_a(m) = 1$, is greater than 0. It indicates that the route enters more PU areas than it exits from the PU areas of channel m . Therefore, the destination node D should be in the PU area of channel m . Then the source node S compares with the sensing results of D . The sensing results of S and D here are long term, similar to the boundary nodes and virtual boundary nodes, which indicate whether S and D are inside or outside the PU areas. If the sensing results of D show that the channel m is always available, which means D is outside the PU area, then the contradictory matching results point out the errors of the sensing results, either by the boundary nodes or the destination node. If a channel m is in L_{out} , it means that the route enters less PU areas than it exits from the areas of channel m . Similarly, S should be in the PU area of channel m . If S 's sensing results show the different results, an error is detected.

Application of Different Physical Layer Models. In our previous discussions, we simplified the physical layer, by assuming that the transmission delay on each node is the same, and calculating the route weight without considering the physical parameters, e.g., the transmission power, or the interference ratio. However, our model can be easily extended and adapted to different physical layer models. The only thing that needs to change is the weighted link length and route length calcu-

Table 5.2: Simulation settings for boundary-based routing.

Number of nodes	[100, 300]
Number of channels	[10, 25]
Number of sectors	4
TX power	23 dBm
Noise power	-98 dBm
SINR threshold	10 dB
Number of PUs	[10, 50]
Operation range of each PU	[300, 500]
Delay for single channel switch	0.1 s

lation. For example, we can apply the SINR model, which calculates the delay of a single link based on power, the link distance, and the interference. One way to extend our model is to define the link weight as the average delay on all channels. Based on Definition 5.3, each channel m on link ab can be associated with a value regarding the estimated delay: if $I_{ab}(m) = 0$, the value associated with m is the estimated delay of ab based on the SINR value; if $I_{ab}(m) = 1$, the value should be higher than the estimated delay, which can be adjusted according to different PU active levels. The weight of link ab is the average value of all channels on it. Then, the weighted route length is the sum of weighted link lengths. Similarly, the route selected here would be the one with the minimum weight.

5.1.6 Simulations

In this subsection, we perform simulations to evaluate the performance of our model. We first introduce the simulation settings. Then we present our simulation results.

Simulation Settings. We randomly distribute nodes in a $2,000 \times 2,000$ unit square. Each node has 4 sectors to send and receive data. We generate a number of PUs, which are randomly active on a certain channel. The operation range of each PU is different. The number of nodes is more than the number of PUs, and this ensures that the boundary of each PU is detected. We randomly choose a source

and a destination. Then, using our approach, the source establishes a route to reach the destination. The channel availabilities are dynamic during the data transmission, because the PUs are set at a predefined probability to be active on a channel. The settings of our simulation parameters are shown in Table 5.2. For simplicity, we set the channel switch delay to be constant.

The three parameters, the number of nodes, number of channels, and number of PUs, are tunable. We compare our model with the shortest path algorithm, which is to find the shortest path from the source to the destination, without considering the channel availabilities. We evaluate the performance of both models from the following aspects:

- Number of hops: simply count the number of links for each route, without considering other factors, e.g., the channel availabilities.
- Total delay: the overall delay considering both channel switching delay and data transmission delay of each session.
- Average route length: calculate the average route length in both models using Definition 5.4.

In addition, we evaluate the influences of adding virtual backbone nodes. We first compare the number of virtual boundary nodes and real boundary nodes by varying the network density, or, the number of nodes in the network. Then, we compare total delay and average route length among models with and without virtual boundary nodes.

Simulation Results. We present our simulation results from the four aspects listed in the above portion.

We set the total number of PUs to 10, and the total number of channels to 10. The number of nodes varies from 100 to 300. We calculate the number of hops of each route under both models. The results are shown in Fig. 5.7. The line labeled as

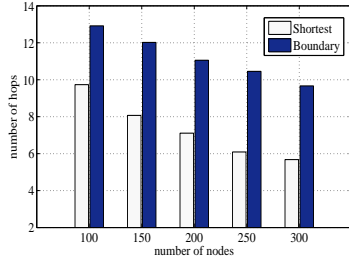


Figure 5.7: Hop number.

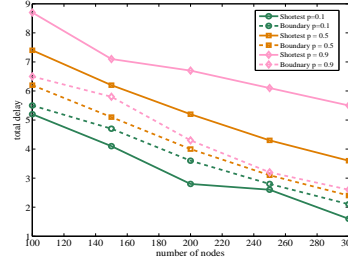


Figure 5.8: Delay.

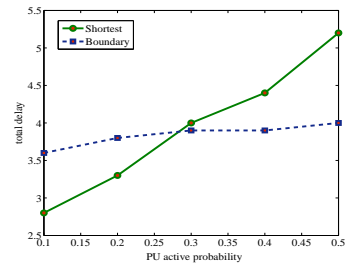


Figure 5.9: Varying PUs.

“Shortest” is the result from using the shortest path algorithm, without considering the channel availabilities. The line labeled as “Boundary” is the result from our model. Obviously, the shortest path algorithm has a lower number of hops than does our algorithm. Moreover, in both models, the average number of hops reduces as the number of nodes increases. This is because the connectivity of the network increases when the number of nodes increases.

We provide the comparison of the total delay for the routes of sessions under both models. We first evaluate the influence of different active PU probabilities on the delay. We set different PU active probabilities (0.1, 0.5, 0.9). We vary the number of nodes from 100 to 300 while keeping the number of channels as 10, and the number of PUs as 10. The results of total delay are shown in Fig. 5.8. We can tell that when the active probability equals 0.5 and 0.9, our boundary-based algorithm is better than the shortest algorithm. When the PU active probability is 0.1, the shortest algorithm achieves shorter total delay than does our model. Therefore, the larger the PU active probability is, the better performance our model will have, compared to the shortest algorithm.

From the above results, we know that the value of the PU active probability, where the two algorithms are close to each other, is between 0.1 and 0.5. Thus, we do more simulations to find the value by varying the active probability from 0.1 to 0.5. We keep the number of nodes as 200, the number of channels as 10, and the number of

PUs as 10. The results are shown in Fig. 5.9. From Fig. 5.9, we can see that the total delay of the two algorithms are the closest when the PU active probability is between 0.25 and 0.3. Based on our threshold-based piggyback scheme, we can set the threshold as 0.3 here. If the PU active probability is below 0.3, the boundary node does not need to piggyback its boundary information. In the following simulations, we set the PU active probability as 0.5.

We then study the influence of the three network parameters; the results are shown in Fig. 5.10. In Fig. 5.10(a), we vary the number of nodes from 100 to 300 while keeping the number of channels as 10, and the number of PUs as 10. The total delay of both models decreases when the total number of nodes increases. Our model achieves about 1.0s less than the model using the shortest algorithm. Besides, in both models, as the number of nodes increases, the total delay increases more slowly. In Fig. 5.10(b), we vary the number of channels from 10 to 25, while keeping the number of nodes as 200, and the number of PUs as 10. The total delay of both models decreases as the total number of channels increases. Under this setting, our model achieves about 20% less total delay than those using the shortest algorithm. In Fig. 5.10(c), we vary the number of PUs while keeping the number of nodes as 200, and the number of channels as 10. The model's total delay increases when the total number of PUs increases. Our model achieves about 20% less total delay than those using the shortest algorithm. In addition, when the number of PUs increases, the total delay of the model using the shortest algorithm increases more quickly. Our model increases much more slowly compared to the shortest algorithm. Therefore, our model is more reliable when facing more dynamic channel availabilities.

We compare the average route length by varying the three tunable parameters: number of nodes, number of channels, and number of PUs, similar to the above settings. In Fig. 5.11(a), the average route length decreases in both models. This is because the number of hops in both models decreases when the number of nodes

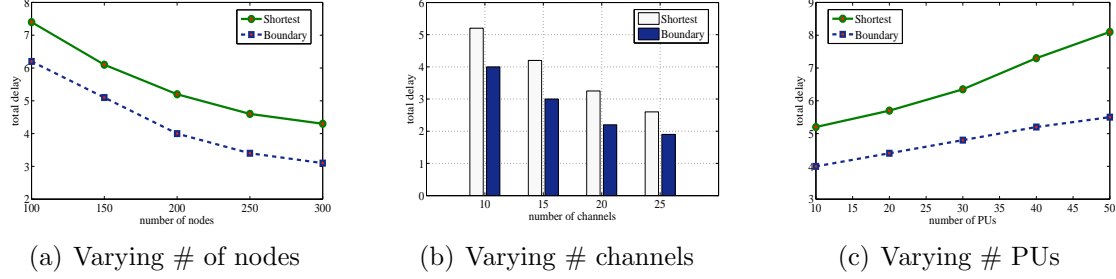


Figure 5.10: Comparison of the total delay by varying network parameters.

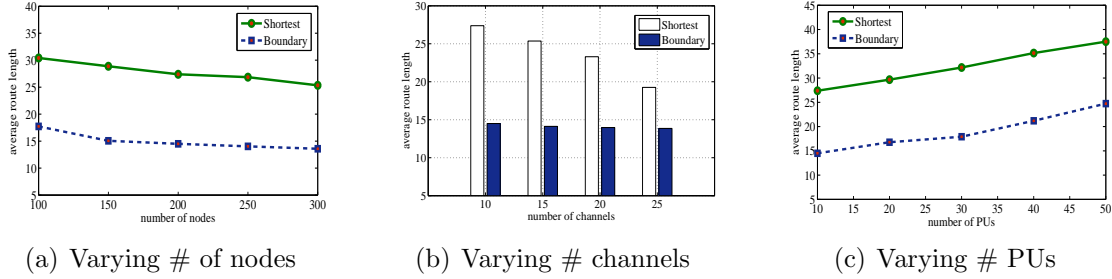


Figure 5.11: Comparison of the average route length by varying network parameters.

increases. The average route length in our model is about 40% less than the shortest path algorithm. In Fig. 5.11(b), the average route length decreases in both models. The average route length using the shortest algorithm is 30% more than in our model. Our model decreases more slightly because it is already close to the minimum value, which equals the number of hops. In Fig. 5.11(c), the average route length increases in both models when the number of PUs increases. This is because, when there are more PUs, more links are within the PU area. In addition, the average route length in our model is about 40% less than the shortest algorithm.

We set the total number of PUs to be 10, the number of operation range as 300, and the active probability of each PU as 0.5. The number of nodes are varied from 60 to 120, considering that the virtual boundary nodes are applied in a sparse network. The other network parameter settings are the same as in the above parts. The average numbers of real boundary nodes and virtual boundary nodes on all the candidate routes from the source to the destination are compared in Fig. 5.12(a). In the model with virtual boundary nodes, the virtual boundary nodes are also counted

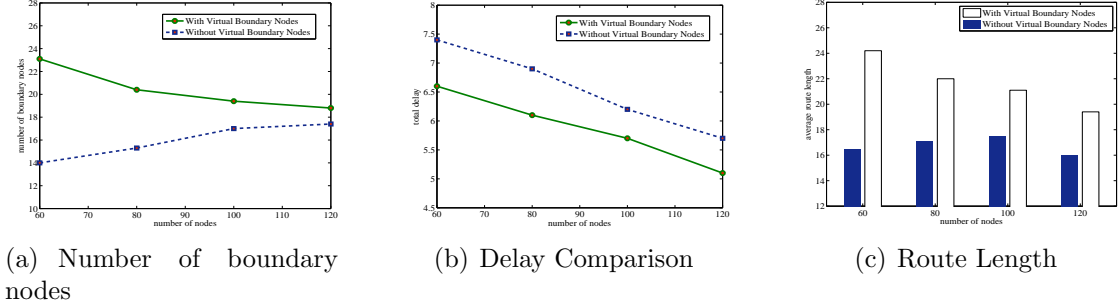


Figure 5.12: Comparison of models with and without virtual boundary nodes.

as boundary nodes. From the results, we can see that the average number of virtual boundary nodes decreases as the number of nodes increases. This is because the average hop distances from the source to the destination decreases, as shown in the above parts. However, the number of boundary nodes in the model without virtual boundary nodes increases when the number of nodes increases. This means that some boundaries are missed in detections without virtual boundary nodes. We compare the total delay in Fig. 5.12(b). The results show that under the same settings, the model with virtual boundary nodes achieves less delay than the one without virtual boundary nodes. In Fig. 5.12(c), the average weighted route lengths are compared. It shows that the model without virtual boundary nodes has less route length than the one with virtual boundary nodes, and it initially increases because some boundaries are not detected. Combined with Fig. 5.12(b), the one with less route length has a larger delay. The reason is that, without virtual boundary nodes, some boundaries are not detected correctly; this causes less weighted route length and, in turn, the missing boundary detections cause longer delay.

5.1.7 Conclusion

In this section, we propose an efficient model for routing in CRNs, which considers the dynamic channel availabilities. We assume that the directional antenna is equipped on each node, aiding each node in determining whether it is a bound-

ary node, and also creating more channel opportunities. Each boundary node is located at the boundary of a PU area. We use the USRP/Gnuradio testbed to prove the differences of the sensing results by boundary nodes in different directions. The boundary nodes help to estimate the channel situation of each optional route, and also the number of links located within a PU area on each route. Nodes on each route piggyback the channel availability and path information. In our model, the piggyback scheme is very efficient, and only causes limited overhead. We give a novel definition of the route path, and propose an effective algorithm for route selection. We analyze the reliability of our algorithm. To improve the feasibility of our model, we consider the situations with imperfect information, and propose the virtual boundary node as well as a threshold-based scheme to overcome the imperfect information. Moreover, in the extensions, we discuss the methods for error detection in sensing results, and how to apply other physical layer techniques in our model. These extensions make our work more reliable and applicable. We perform numerous simulations to testify the performance of our model.

5.2 Forwarding Node Set Selection

A very promising solution to the routing problem in CRNs is the opportunistic routing [76–78], since nodes in such opportunistic routing protocols do not stick to a particular route. Instead, a sender broadcasts its data. Nodes which hear the transmission and are closest to the destination will forward the data. This scheme is particularly useful for routing in CRNs, considering the special interference and channel dynamics. In CRNs, different from other wireless networks, the interference to a single link can come from PUs and SUs. If the interference is caused by SUs, the interfered nodes can compete for channel access. However, if the interference is caused by PUs, the nodes that have been interfered with cannot compete with PUs, but quit from that channel. This causes links in CRNs more vulnerable and taking

a very long time to recover once interfered by PUs. Therefore, routes consisting of single links in CRNs are unreliable when facing the unpredictable PU activities. Since the opportunistic routing does not necessarily rely on a single route, which is more reliable under the dynamic channel availabilities, it can potentially be very useful if applied in CRNs. With tremendous works on opportunistic routing problems in other wireless networks, it brings up a question: are the previous approaches of opportunistic routing directly applicable on CRNs?

In fact, the answer is negative. One of the problems is the forwarding node (FN) set selection of opportunistic routing in CRNs. The FN set selection in traditional wireless networks [79–82] usually relies on the distance to destinations, link qualities, estimation of delivery probabilities, and so on. There are no PUs in traditional wireless networks, which means nodes in these networks do not need to stop using channels immediately for active PUs. Therefore, the FN set selection scheme in traditional wireless networks has no consideration for PUs.

In addition, due to the unpredictability of PUs' activities, if nodes in a selected FN set are affected by PUs, and these PUs stay active for a long time, the FN set needs to be adjusted for its reducing performance decrement. Therefore, an adjustment scheme for the FN set in CRNs is necessary. This is different from traditional networks, since nodes in traditional networks can compete for the channels, and there are no privileged nodes. Thus, a new adjustment scheme is required to ensure the performance of CRNs in the long run.

In this section, we propose a novel FN set selection model in CRNs, which aims at meeting the requirements regarding the delivery rate. We consider both the reliability and the co-channel interference, and define the weight of each candidate FN. For different performance considerations, we propose three algorithms to select the FN set for each node: a basic greedy algorithm, a greedy algorithm with one backtrack, and a maximum weighted independent set (MWIS) algorithm.

5.2.1 Problem Formulation

We consider that, in a CRN with node set $\{u, v, w, \dots\}$, the total available channel set is M . There are also a number of PUs with random probabilities of being active. When the PUs become active on certain channels, the nodes within the PUs' interference area have to quit the channels. For a node u , we use M_u to denote the current available channel set of u . The sender and receiver have to tune to the same channel to communicate. We assume there is a CCC in our model. In addition, the SINR(signal to interference-plus-noise ratio) from the sender node to the receiver has to be large enough for a successful transmission.

Assume that the opportunistic routing is applied for the data transmission. We are not limited to a specific opportunistic routing protocol. Our model is to add the consideration of specialities in CRNs to the FN selection. Suppose we apply SOAR [78] here, and each node maintains a routing table containing (destination, default path, and FN set). Our goal is to choose the FN set from the neighbor set of each node, and ensure that the delivery rate to the destination node satisfies the predefined threshold. In the following parts, we formulate the delivery probability for a single link based on the above constraints, and provides the assurance for the dynamic channel adaptability.

For a single link between a pair of nodes with distance d transmitting on channel m ($m \in M$), the average SINR value is denoted as $\Gamma(d, m)$. We use the Rayleigh fading model to estimate small scale fading here[79, 83]. When channel m is available, which means the PUs on m are inactive, the probability of bit error over a distance d is[84]:

$$p_e(d, m) = \frac{1}{2} \left(1 - \sqrt{\frac{\Gamma(d, m)}{1 + \Gamma(d, m)}} \right). \quad (5.2-9)$$

In our model, we assume each packet has a constant length of L . The probability of PUs, whose interference areas contain this link, being inactive on channel m is

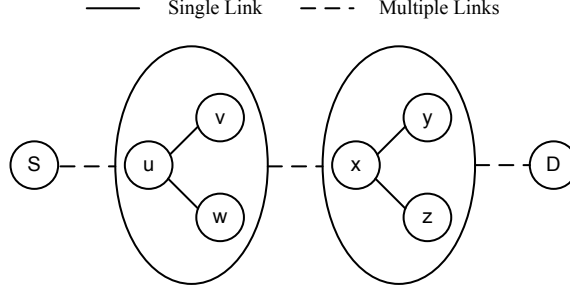


Figure 5.13: The FN adjustment is needed for x , rather than u .

$H(m)$. We use uv to denote the single link from node u to v , and d_{uv} to denote the distance between u and v . The channel set that can be used for transmission is $M_{uv} = M_u \cap M_v$. The probability of a packet being transmitted successfully over link uv can be written as:

$$p(d_{uv}) = 1 - \prod_{m \in M_{uv}} (1 - H(m) \cdot (1 - p_e(d, m))^L). \quad (5.2-10)$$

The $p(d_{uv})$ is related to both the SINR constraint, and the PU protection requirements in CRNs. It is only an estimation because $H(m)$ is a historically statistical value, and sometimes it is not precise. In addition, the interference of other links to uv is not static, since they may switch among different channels during the transmission. However, it does provide a good insight for the quality of a single channel, which will be one of the factors that are considered for FN set selection.

The PUs in CRNs are likely to be suddenly active during the data transmission, which may cause the SINR of some links to fall below the required threshold. Once it happens, the affected links can switch to other available channels, and continue transmission. However, when none of the channels of a link can meet the SINR requirement, the link will become broken. During this process, the delivery rate to the destination is likely to decrease and fall below the required threshold. Considering this situation, the FN adjustment is inevitable, as to adapt to the dynamic channel environment in CRNs.

The influences caused by suddenly active PUs can be studied through ACKs. There are two types of ACKs for different uses[78]. One is the per-hop ACK, which is used to confirm having received the packets among one-hop neighbors. Another is the end-to-end ACKs, which is used for rate control, and is sent by the destination after receiving a batch of packets. Our model does not need to change the format of ACKs, or the schema of per-hop ACKs and end-to-end ACKs, but uses the end-to-end ACKs to determine the appearance of active PUs and to adjust the FN set for affected nodes.

For a single node, when the received end-to-end ACKs from the destination become fewer, and the rate control does not improve the number of ACKs, it needs to decide whether the FN adjustment is necessary. It is possible that the FN adjustment is not necessary on this node, but on some of its downstream nodes to the destination. For example, in Fig. 5.13, suppose that some PUs suddenly become active and cause the decrement in the delivery rate of the FN set $\{y, z\}$ of x . This may, in turn, result in a decrease in the number of ACK's received by u . Node x should adjust its FN set, rather than u , although both of their received end-to-end ACKs become fewer. Therefore, not only is the FN adjustment necessary, but also finding the nodes that really need the FN adjustment is crucial.

From the above discussions, it is impractical to find an optimal FN set for each node, due to the dynamic channel environment. We propose a practical and effective FN set selection model, which considers both the co-channel interference among links within the same interference area, and the reliability when PUs become active during transmission. Also, an efficient FN adjustment scheme is necessary to ensure the delivery performance of opportunistic routing in CRNs.

5.2.2 FN Set Selection Model

For a node u , the overview of its FN set selection is :

- Node u selects a subset from its neighbor set;
- Having the FN set selected, node u sends the coded packets on the channel coordinated using CCC;
- Once PUs appear and cause the channel being used by node u to become inactive, u would switch to another available channel and continue transmitting.

Here are some illustrations. The FN set selection algorithm is described in the following parts. We propose three algorithms, and each of them is suitable for different scenarios. Nodes in the FN set of u will receive packets from u using the original channel as u . The FN set selection algorithms do not depend on which channel is used by u for transmission. The channel selection and coordination among multiple nodes for packet broadcasting, which can be conducted by transmission performance estimation and controlling message exchanges, is beyond the scope of this paper. When more PUs become active, and the channel switching cannot meet the delivery rate requirement, the FN set of some nodes will be adjusted.

In the following parts, we first prioritize the candidate FN nodes in the neighbor set by defining their weight, considering the channel dynamics in CRNs. Then, three FN set selection algorithms are proposed and compared.

Forwarding Node Prioritization The delivery probability estimation of a packet over a single link is given in Eq. 5.2-10. From Eq. 5.2-10, it is obviously that the main factors on the delivery probability are:

1. the interference, $I(m)$, from other links within the interference area of uv that are also working on channel m ;
2. the probability of PUs, $H(m)$, whose interference area contains link uv , that are inactive on channel m .

Therefore, both factors should be taken into consideration when selecting FNs from neighbor nodes.

For simplicity, we assume that the interference range of a single link uv is two-hop, which includes all adjacent links of uv . This means that $I(m) = \infty$ when there is an adjacent link also using m ; $I(m) = 0$, otherwise. This assumption is not a requisite for our model. Other interference estimation methods can be easily applied here.

Assume that u needs to select FN from its neighbor set, N_u . For a node $v \in N_u$, M_u is the available channel set of node u . We use M_{uv} to denote the channels available on link uv , and $M_{uv} = M_u \cap M_v$. Considering the interference from adjacent links, for any channel $m \in M_{uv}$, the conflict probability of adjacent links choosing the same m as uv is defined as:

$$C_{uv}(m) = \sum_{w \in N_v} \frac{1}{|M_{uw}|} E_{uw}(m) + \sum_{w \in N_u} \frac{1}{|M_{vw}|} E_{vw}(m), \quad (5.2-11)$$

where $E_{uv}(m)$ is a step function with a value of 1 when $m \in m_{uv}$ and is 0 otherwise. Then, we define “receiving ability” regarding a neighbor node v of node u .

Definition 5.7. For $\forall v \in N_u$, the receiving ability of v from u is the following:

$$R_{uv} = \sum_{m \in M_{uv}} \frac{1}{C_{uv}(m)}. \quad (5.2-12)$$

This definition considers the interference from the adjacent links. Node v , with more channels that are less likely to be used by adjacent links, has a better value of R_{uv} . Moreover, node v that has more available channels to be used for receiving from u is more reliable. When PUs nearby suddenly become active, it can switch to one of the other channels, and continue transmission.

However, it is not enough to prioritize each neighbor node using the notion of receiving ability. This is because node v with a better value of R_{uv} is likely to have a worse ability of forwarding to the next-hop nodes of v . When node u decides to choose v as its FN, it also needs to estimate the forwarding ability of v . Assume

Algorithm 11 Basic Greedy Algorithm to Calculate F_u of u

- 1: N'_u is the list to store the FN candidates
 - 2: **for** $v \in N_u$ & v is smaller of ETX to the destination than u **do**
 - 3: **if** ETX of v to the destination $< \alpha$ **then**
 - 4: Calculate W_{uv} using Eq. 5.2-13
 - 5: Insert v to N'_u
 - 6: **while** N'_u is not empty **do**
 - 7: Set v as the node with the max W_{uv} in N'_u
 - 8: **if** $d_{vw} > \sigma, \forall w \in F_u$ **then**
 - 9: $F_u = F_u + \{v\}$
 - 10: $N'_u = N'_u - \{v\}$
-

that u has the two-hop information when it performs the estimation. We have the following definition of the weight of node v to be chosen as a FN of u :

Definition 5.8. For $\forall v \in N_u$, the weight of v to be selected as a FN of u is:

$$W_{uv} = R_{uv} \left(\sum_{w \in N_v} |M_{vw} - M_{uw}| \right). \quad (5.2-13)$$

$|M_{vw} - M_{uw}|$ is the number of elements left in M_{vw} after removing the elements in M_{uw} from M_{vw} . With the same receiving ability, the nodes that have more non-conflict channels with u and more links to forward the packets will have a larger weight.

Here, if v can be selected as the FN of u , it should be closer in terms of its proximity to the destination. The proximity here is not measured by physical distance, but the ETX metrics as in [85]. Additional constraints of ETX can be applied here, e.g., the four constraints in SOAR [78]. Our FN node set selection algorithm can be combined with other FN selection algorithms in terms of different proximity metrics.

FN Set Selection Algorithms. Intuitively, the easiest way to select a FN set is to apply the greedy algorithm. For a node u , the basic greedy algorithm of choosing its FN set, F_u , is described in Algorithm 11. The FN set is selected from the downstream neighbors of u , which are closer to the destination than u , in terms of ETX metrics,

Algorithm 12 Greedy Algorithm With Backtrack List Maintained to Calculate F_u of u

- 1: L_B is the list to store the backtrack nodes
 - 2: N'_u is the list to store the FN candidates
 - 3: **for** $v \in N_u$ & v is smaller of ETX to the destination than u **do**
 - 4: **if** ETX of v to the destination $< \alpha$ **then**
 - 5: Calculate W_{uv} using Eq. 5.2-13
 - 6: Insert v to N'_u
 - 7: **while** N'_u is not empty **do**
 - 8: Set v as the node with the max W_{uv} in N'_u
 - 9: **if** $d_{vw} > \sigma, \forall w \in F_u$ **then**
 - 10: $F_u = F_u + \{v\}$
 - 11: Set v' the node with the second max $W_{uv'}$ in N'_u
 - 12: **if** $d_{v'w} > \sigma, \forall w \in F_u$ **then**
 - 13: $L_B = L_B + \{v'\}$
 - 14: $N'_u = N'_u - \{v\}$
-

and also satisfy the ETX threshold α . The ETX here of a single link is calculated as $\frac{1}{(1-p_r) \times 1-p_f}$, where p_r and p_f denote the loss probabilities of the link in the forward and reverse directions, respectively. The ETX metrics have been proved useful in [86]. Also, we set the distance constraint σ on nodes in F_u . There are two reasons for setting σ on the FN set. First, nodes in close geographical locations share similar channel availabilities in CRNs. If two nodes are very close together, it is possible that they choose the same channel for data transmission, which would cause co-channel interference. Second, when a PU becomes active, nodes that are too close to each other may be affected together, and they all need to switch channels. Therefore, we require nodes in the the FN set to satisfy the minimum distance requirements. Having the ETX and distance constraints satisfied, the node with the maximum weight is selected, and it will be added to F_u .

The basic greedy algorithm is very straightforward and easy to implement. However, it is likely that the FN set selected by the greedy algorithm is too small. For example, under some circumstances, once the node with the maximum weight is selected, there is only one more, or even no nodes that satisfy both the ETX and

distance requirements, which makes the size of the FN set only contain one or two elements. This would harm the routing performance since the forwarding ability of FN set is harmed. Also, if the application scenario has other requirements on the FN set, the greedy algorithm is very likely to fail in satisfying the requirements. Therefore, the greedy algorithm is likely to be ineffective. To deal with this situation, we provide one backtrack scheme for the greedy algorithm.

During every loop in which one node is selected into the FN set by the greedy algorithm, we keep a backtrack node for it. For example, if node v is selected into the FN set, we would select a node v' with the second largest weight. If the final result of the FN set selected by the greedy algorithm cannot meet the other requirements (e.g., the size requirement of FN set), our backtrack scheme will go back one step, replace one element every time, and rerun the greedy algorithm from that point. The details of the greedy algorithm for selecting F_u for node u with backtrack list L_B maintained is in Algorithm 12. The initial FN set selection process is similar to the greedy algorithm, except that a L_B list is maintained to store the backtrack node for every node in the initial F_u . Here, some overlap is possible between the two lists, F_u and L_B .

The backtrack algorithm is in Algorithm 13 with the size requirement τ . Similarly, it can be extended for other requirements other than τ . When the F_u cannot meet the requirement τ , the FN set would be updated by removing all the nodes with weights less than the one pointed by ptr_1 . Also, the ptr_1 and ptr_2 are used to maintain the backtrack node. Then the greedy algorithm will be applied to select new nodes to the FN set from the remaining nodes with weights less than that of the node pointed by ptr_2 . The process will continue until the FN set meets the constraint of τ . We only maintain one backtrack list here. Of course, more backtrack lists can be maintained, if one cannot find the appropriate FN set. The complexity of this backtrack scheme is low, since only one node is maintained for each node in the FN set.

Algorithm 13 Backtrack Algorithm With Size Constraint τ

- 1: Pointer ptr_1 points to the end of the F_u
 - 2: Pointer ptr_2 points to the end of the L_B
 - 3: N'_u is the list to store the FN candidates
 - 4: **while** $|F_u| < \tau$ **do**
 - 5: Remove all nodes with weights less than that of the node pointed by ptr_1
 - 6: Replace the node of F_u that ptr_1 points to with the node pointed by ptr_2 in L_B
 - 7: **for** $v \in N_u$ & v is smaller of ETX to the destination than u **do**
 - 8: **if** ETX of v to the destination $< \alpha$ **then**
 - 9: Calculate W_{uv} using Eq. 5.2-13
 - 10: **if** $W_{uv} <$ the weight of the node pointed by ptr_1 **then**
 - 11: Insert v to N'_u
 - 12: **while** N'_u is not empty **do**
 - 13: Set v as the node with the max W_{uv} in N'_u
 - 14: **if** $d_{vw} > \sigma, \forall w \in F_u$ **then**
 - 15: $F_u = F_u + \{v\}$
 - 16: $N'_u = N'_u - \{v\}$
 - 17: Move ptr_1, ptr_2 one step forward
-

The basic greedy algorithm and the greedy algorithm with one back track cannot ensure that the selected FN set is the one with the maximum weight. Here, under the same ETX and distance constraints, we propose another algorithm to give the optimal result as of the overall weight. For the neighbor set N_u of node u , we construct a graph $G_u(\sigma)$ from nodes that satisfy the ETX constraints α , defined as follows:

Definition 5.9. Given node u , its neighbor set N_u , and the distance threshold σ , we define a graph $G_u(\sigma)$, where

1. v is a vertex in $G_u(\sigma)$, iff $v \in N_u$, v is smaller of ETX to the destination than u and satisfies ETX constraint α ;
2. an edge exists between two vertices, v and w , in $G_u(\sigma)$ iff $d_{vw} > \sigma$,

Based on the weight definition in Eq. 5.2-13, the FN set selection does not rely on any specific channel used by each node. The goal here is to find the maximum weight set with the ETX and distance constraints. We define the independency here on the distance threshold σ , and convert the FN set selection to find the Maximum

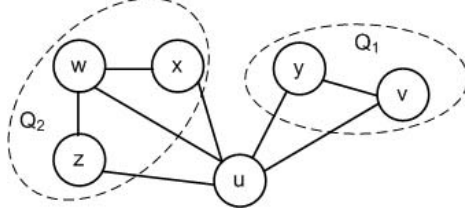


Figure 5.14: An example of the modules for node u .

Weighted Independent Set (MWIS) of $G_u(\sigma)$.

The MWIS problem is a well-known NP hard problem. We adopt a recursive approach for MWIS calculation, based on module decomposition[87]. A module is defined as:

Definition 5.10. Given a $G_u(\sigma)$, suppose U is a subset of the vertex in $G_u(\sigma)$. For a node v , which is a vertex of $G_u(\sigma)$ and $x \notin U$, x “distinguishes” U if x has both a neighbor and a non-neighbor in U . U is a *module* if it is indistinguishable for the vertices outside U .

We use $\{Q_k\}$ to denote the set of modules in $G_u(\sigma)$. An example is shown in Fig 5.14. Suppose $\{x, y, z, v, w\}$ are u ’s neighbors, and construct $G_u(\sigma)$. The node x distinguishes the node set $\{z, w\}$, since x has both a neighbor and a non-neighbor in $\{z, w\}$. x cannot distinguish the node set $\{y, v\}$ since neither y nor v is x ’s neighbor. There are two modules in this graph, which is $Q_1 = \{y, v\}$ and $Q_2 = \{z, w, x\}$.

Having the module defined, the MWIS algorithm contains two steps: 1) decompose the $G_u(\sigma)$ into different modules; 2) recursively find the MWIS in each module to get the MWIS in $G_u(\sigma)$. The following are implementations of the two steps.

Definition 5.11. Given a $G_u(\sigma)$, based on the fact whether it is connected or not, the module decomposition process is:

- if $G_u(\sigma)$ is disconnected, it can be divided into modules $\{Q_k\}$, which are connected components;

Algorithm 14 MWIS Algorithm to Calculate F_u of u

- 1: Set $G_u(\sigma)$ as empty
 - 2: **if** $G_u(\sigma)$ only has one vertex, v **then**
 - 3: Return $F_u = \{v\}$ and stop
 {Construct the graph $G_u(\sigma)$ }
 - 4: **for** every $v \in N_u$ & v is smaller of ETX to the destination than u **do**
 - 5: Calculate W_{uv} using Eq. 5.2-13
 - 6: Add v to the vertex set of $G_u(\sigma)$
 - 7: **for** every w that is a vertex in $G_u(\sigma)$ **do**
 - 8: **if** $d_{wv} > \sigma$ & ETX of v to the destination $< \alpha$ **then**
 - 9: Add edge wv to $G_u(\sigma)$
 - 10: Divide $G_u(\sigma)$ into $\{Q_k\}$ using Definition 5.11.
 - 11: **for** every $Q_k \in \{Q_k\}$ **do**
 - 12: Set $G_u(\sigma) = Q_k$
 - 13: Rerun from 2, with output denoted as F_u^k
 - 14: $F_u = F_u \cup F_u^k$
-

- if the complement graph of $G_u(\sigma)$, denoted as $\overline{G_u}(\sigma)$ is disconnected, $\overline{G_u}(\sigma)$ can be divided into $\{Q_k\}$, which are connected components;
- if both $G_u(\sigma)$ and $\overline{G_u}(\sigma)$ are connected, divide $G_u(\sigma)$ into maximal modules $\{Q_k\}$.

Obviously, for any module Q_k in connected $G_u(\sigma)$, there exists no node u outside Q_k that has both a neighbor and a non-neighbor in Q_k . The maximal modules in the third case are all pairwise disjoint.

Having the decomposition scheme, the recursive MWIS algorithm for finding FN set with the distance threshold σ is shown in Algorithm 14. The first part is to construct the $G_u(\sigma)$ based on the threshold σ . The second part is to divide the original $G_u(\sigma)$ into different modules or components $\{Q_k\}$. The third part is to recursively find the MWIS in each Q_k and return the MWIS of $G_u(\sigma)$. Similar to the greedy algorithm, the dynamic adjustment of F_u is inevitable due to the PUs' activities. The details are discussed in the next part.

5.2.3 Performance Analysis

We analyze the complexity of the above three algorithms in this part. For node u , the number of its neighbor nodes is $|N_u|$, denoted as N . Obviously, the complexity of the greedy algorithm is $O(N \log N)$. For the other two algorithms, we have the following theorems:

Theorem 5.12. *For node u with N neighbor nodes, the complexity of the greedy algorithm with one backtrack to find the F_u is $O(N^3 \log^2 N)$.*

Proof. The worst case is that each node in the initial selected FN set needs to be replaced with its back track node; then, the complexity is $N \cdot \log N \cdot (\sum_{1 \leq i \leq N} i \log i)$:

$$\begin{aligned} N \cdot \log N \cdot \left(\sum_{1 \leq i \leq N} i \log i \right) &\approx N \cdot \log N \int_1^N i \log i \, di \\ &= N \cdot \log N \cdot \left(\frac{1}{2} \cdot N^2 (\log N - 0.5) \right) = O(N^3 \log^2 N). \end{aligned}$$

□

Theorem 5.13. *For node u with N neighbor nodes, if the complexity of finding MWIS for the subgraphs of $G_u(\sigma)$ is $O(Np)$, where p is a constant and $p \geq 1$, then the complexity of finding F_u using the MWIS algorithm is $O(N^2)$.*

Proof. If the complexity of finding MWIS for the subgraphs of $G_u(\sigma)$ is $O(Np)$, where p is a constant and $p \geq 1$, as proved in [87], the complexity of finding F_u is $O(Np + q)$, where q is the number of edges. Since $q \leq O(N^2)$, the complexity of MWIS is $O(N^2)$. □

Therefore, the complexity of the MWIS algorithm depends on the complexity of solving the subgraphs, or modules. Moreover, the worst case of the greedy algorithm with one backtrack has a higher complexity than the greedy algorithm. We will

compare the performance of the selected FN set by three algorithms, as well as their reliability in our simulations.

5.2.4 FN Adjustment

In CRNs, when PUs become available, the affected nodes may choose to select other channels and continue transmission. It seems that the FN adjustment is not necessary since the affected FNs have other channel options. However, this does not always work; since sometimes, the number of available channels can be very small when there are too many PUs, or other nodes take up a lot of channel resources.

Our FN adjustment scheme makes use of the end-to-end ACKs, like in [78]. The end-to-end ACKs are sent by the destination, and enable each node to know the frequencies and the packets that are received by the destination. For node u , it can tell the delivery rate or the routing quality from recording the frequencies of the end-to-end ACKs, regarding its packet. We use f_u to denote the frequencies of end-to-end ACKs received by u over a constant time period.

In CRNs, the sudden appearances of PUs would harm the channel availabilities. When the channel availabilities of some nodes in a FN set are affected, the f_u will reduce, for every upstream u of the affected FN set. For example, in Fig. 5.13, when the FN set of x are affected by the suddenly active PUs, the f of the upstream nodes of x (e.g., u , v , w) will be affected. Obviously, some adjustment is necessary. Instead of having all affected nodes make adjustments, our FN adjustment scheme only requires a limited number of nodes that need to change the FN set and, therefore, reduces the cost of adjustments under dynamic channel environment. We set λ as the threshold of f . For node u , when f_u falls below λ , the adjustment process will start, as described below:

1. If the channel currently used by u is affected by the suddenly appeared PUs, u switches to another channel and continues transmitting. If no other channel

can be found for transmission, u will send a *failure* signal over the CCC to its one hop upstream node u' , where $u \in F_{u'}$;

2. If u is not affected by PUs, or successfully continues transmission by channel switching, it keeps listening on the CCC for a *failure* signal. If a *failure* signal is received by u , it will rerun Algorithm 14. If f_u still cannot reach λ over a time period Δt , u will send the *failure* signal to u' , where $u \in F_{u'}$;
3. If u is neither affected by PUs, nor does it receive any *failure* signal, it will keep listening and transmitting.

In the above process, step 1 deals with the nodes that are affected by the suddenly active PUs. They only send the *failure* signal when the transmission cannot be fixed by channel switching. Step 2 deals with the situation in which the channels of the FN nodes are affected by PUs and cannot be fixed by channel switching. Step 3 is for nodes that are not affected by PUs, and their FN set is not affected either; they will keep transmitting, and continue listening over CCC.

Take Fig. 5.13 as an example. Nodes v and w are the FNs of u . Nodes y and z are the FNs of x . Suppose both x and y are affected by the suddenly active PUs nearby, which cause the f of nodes x, y, u, v, w to be unable to meet the requirement λ . First, x and y will find another channel to switch to. Suppose y cannot find any other appropriate channels; it will send a *failure* signal to x . Suppose x finds another channel and continues transmission. When x receives the *failure* signal, it will run Algorithm 14 and reselect its F_x . Obviously, node y would not be selected to F_x . Node z may or may not be selected to F_x . If f_x cannot reach λ after Δt , it will stop and send a *failure* signal to its one-hop upstream node. Nodes u, v, w are not affected by PUs. They will do nothing but keep transmitting and listening over the CCC for a *failure* signal.

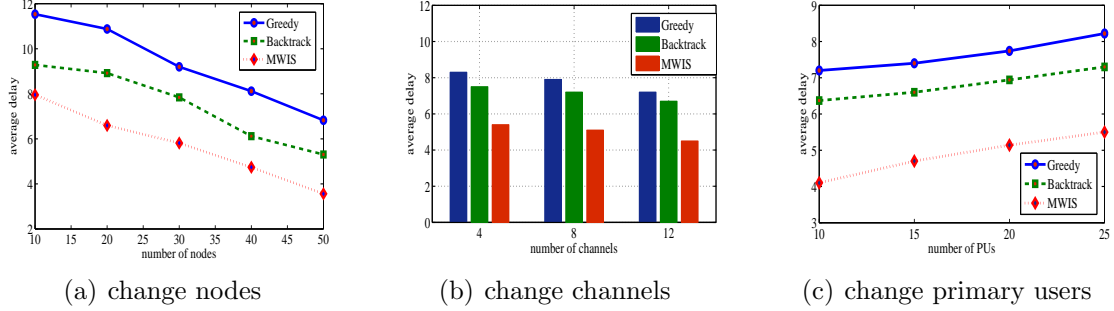


Figure 5.15: Comparison of delay under different network environment parameters.

5.2.5 Simulations

We randomly distribute nodes in a 200×200 unit square. There are three network parameters: the number of nodes, the number of total channels, and the number of PUs. The total number of node varies from 10 to 50. The communication range of each node is $[50, 70]$. Each node has its own set of available channels, depending on the position of PUs. There are $[4, 12]$ total channels, and the total number of PUs ranges from 10 to 25. Each PU has a probability 0.5 of being active at the beginning of each time slot, and occupies one channel. We assume each packet is transmitted at the beginning of each time slot. For a single link, the time cost to transmit a packet on an available channel is one time slot. We assign the loss rate of a single link as $[0.2, 0.8]$. By varying the three network parameters, we compare the following two performance metrics:

- Average delay: Given constant traffic requirements, the average value of delay used by each algorithm to reach the destination node;
- Number of FN adjustments: The average number of FN adjustments by each node, due to the dynamic activities of PUs.

Average Delay. We compare the average delay by varying three different network parameters. In Fig. 5.15(a), the number of nodes varies from 10 to 50. The number of total channels is set to be 10, and the number of PUs is 15. The three lines denote the

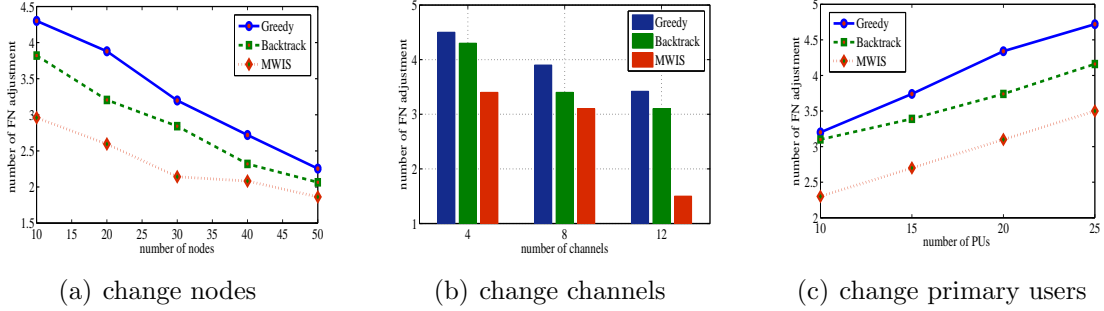


Figure 5.16: Comparison of FN adjustments under different network environment parameters.

greedy algorithm, greedy algorithm with one backtrack, and MWIS algorithm. The delay is measured in seconds. The results show that MWIS has the least delay among the three algorithms, while the greedy algorithm has the most delay. The delay of all three algorithms decreases as the number of nodes increases. This is because more nodes can be selected in the FN set, given a certain traffic.

In Fig. 5.15(b), the average delay is compared by varying the number of total channels in the network. The total number of channels is varied from 4 to 12. The number of nodes is 25, and the number of PUs is 15. Again, the MWIS algorithm achieves the least delay among the three, and greedy algorithm with one backtrack is the second best. The delay of all three algorithms decrease as the number of channels increases. This is because more channels bring more channel opportunities for each link, and make each link more stable.

In Fig. 5.15(c), we vary the total number of PUs in the network, and compare the delay. The number of PUs varies from 10 to 25. The number of nodes is set as 25, and the number of total channels is 10. The results show that MWIS takes the least delay, and the greedy algorithm takes the most delay. The delay of all three algorithms increase as the number of PUs in the network increases. This is because, the more PUs there are, more interference there would be.

Overall, MWIS has the best performance of average delay among the three algorithms, while the greedy algorithm takes the largest delay. The MWIS achieves

almost 30% less than the greedy algorithm in delay. Moreover, the three algorithms have a similar trend when varying the network parameters.

Number of FN adjustments. The number of FN adjustments is calculated as the average number of FN sets of each node. The PUs randomly become active at the beginning of each slot, which causes the FN adjustments to become necessary. In Fig. 5.16(a), the number of FN adjustments is measured by varying the number of nodes in the network. The number of nodes varies from 10 to 50. The number of channels is 10, and the number of PUs is 15. When the number of nodes increases, the average number of FN adjustments decreases for all three algorithms. Among the three algorithms, MWIS costs the least number of FN adjustments, while the greedy algorithm takes the highest number of FN adjustments.

In Fig. 5.16(b), we evaluate the number of FN adjustments by changing the number of total channels from 4 to 12. The number of nodes is kept as 25, and the number of PUs is 15. The simulation results show that MWIS needs the least number of FN adjustments, and the greedy algorithm has the second best performance. All three algorithms need fewer numbers of FN adjustments when the number of channels becomes more.

In Fig. 5.16(c), the number of FN adjustments is measured by changing the number of PUs from 10 to 25. The number of nodes is set as 25, and the number of channels is 10. The results show that MWIS requires the least number of FN adjustments. When the number of PUs increases, the number of FN adjustments increases for all three algorithms.

5.2.6 Conclusion

We consider the FN set selection problem in CRNs under the opportunistic routing. Three algorithms are proposed, based on the weight definition considering the channel dynamics of CRNs. Considering the dynamic channel availabilities in CRNs,

we develop a FN adjustment scheme, considering the characteristics of channels availabilities in CRNs. The simulation results show the performance of our algorithms.

5.3 Chapter Conclusion

In this chapter, we study the routing problems in CRNs. The routing problems in CRNs are different from traditional wireless networks due to the appearances of PUs. When PUs become active, some links are likely to be broken immediately, which results in the decrement of the routing performances. Our work mainly lies in two aspects. One is the boundary assisted routing. The other one is the FN set selection in CRNs.

We make use of boundary nodes to piggyback information, with low overhead. We redefine the route length in a creative way, which considers the channel availability, route stability, and the delay. Based on our defined route length, we give the algorithm for the route selection. Moreover, we propose schemes for situations with imperfect information, e.g., missing boundary detection and imprecise sensing results, which improve the feasibility and reliability of our model.

We consider the FN set selections in CRNs under opportunistic routing. We define the weight of each candidate FN by taking the dynamic channel availabilities and co-channel interference into account. Three algorithms are proposed for the FN set selection, aiming at different performance requirements with varied complexities. We also develop an efficient approach for FN adjustment considering PUs in CRNs.

The discussions on the routing protocols have shown the complexity under the highly dynamic and unpredictable environment in CRNs. In the next chapter, we will study the infrastructure design. A general infrastructure will bring many benefits to CRNs, e.g., the channel management and routing issues.

CHAPTER 6

INFRASTRUCTURE DESIGN FOR CRNS

As discussed in previous chapter, the high dynamics of channel availability makes it costly to collect information from other nodes and construct a routing path. 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. In this chapter, we will discuss our works for the architecture design in CRNs to facilitate the data transmissions.

6.1 Virtual Backbone Construction

In the following sections, we propose a novel approach for virtual backbone construction in a CRN without relying on a CCC. 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 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 would calculate 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, where each node communicates with the backbone node

only. Hence, with our approach, both the reliability and throughput are improved than in a CRN without a backbone, and a traditional wireless network with backbone. To avoid cochannel interference among different links, we propose an algorithm that chooses a different transmission channel for each simultaneously communicating node pair.

6.1.1 Preliminaries

To solve the problems caused by channel dynamics in CRNs, we can make use of the virtual backbone structure [88] in traditional wireless networks. 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. 6.1, each node is either a backbone node or attached to a backbone node. A_1 denotes an area, which includes the backbone node and nodes attached to it. 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 $A_1 \rightarrow A_2 \rightarrow \dots \rightarrow A_k$. Moreover, the virtual backbone can solve the unstable link problem. This is 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.

However, the virtual backbone construction in traditional wireless networks [89–91] relies on a control channel to exchange extensive control information during the virtual backbone construction. In CRNs, due to the dynamic availability of chan-

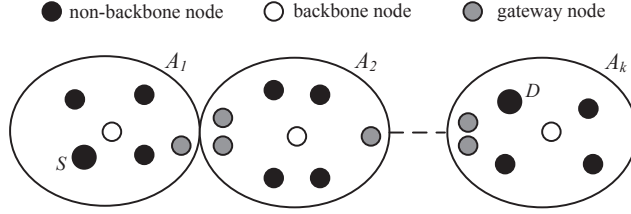


Figure 6.1: Example of the end-to-end transmission using virtual backbone.

nels, it is 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 attack and congestions. Furthermore, for the spectrum authorities to allocate a static band as the control channel for CRNs, it is often involved with international negotiations and, hence, is very time consuming. Therefore, we need to find an efficient way for nodes in CRNs to form a virtual backbone without a CCC.

6.1.2 Problem Formulation

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. We divide the network into a set of square cells, $C = \{c_k | k = 1, 2, 3, \dots\}$. 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 the equipped GPS waveform, 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. Specifically, 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 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 in CRNs is developed. We will introduce the above three phases in the following three subsections.

6.1.3 Self-Organization

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 6.1. 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 for a passive node to listen to. We call such a channel for each node as an *initial home channel* (IHC),

Algorithm 15 $Pr01(i, M(c_k))$, 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 k th channel in \mathcal{Q} }
 - 6: $\mathcal{Q} = \mathcal{Q} \setminus \{q\}$ {Remove q from \mathcal{Q} }
 - 7: **until** $q \in M_i$
 - 8: Return q {Selected IHC}
-

which is used for exchanging control messages initially. We describe how to select IHC next.

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 $Pr01(i, M(c_k))$ for node i in cell c_k to obtain its IH_i is shown in Algorithm 15.

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 a better method to 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:

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

The channel segment needs to be reused under some circumstances. The constraint

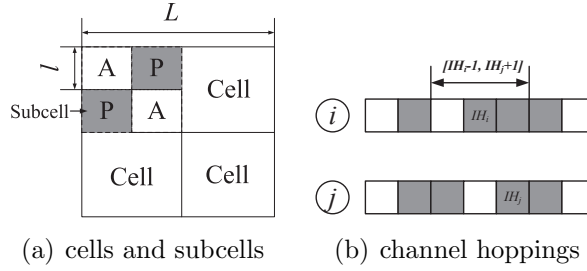


Figure 6.2: Example of self-organization

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.

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 remaining. 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.

To determine the state of each node, we apply the subcell concept here.

Definition 6.2. 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. 6.2(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.

For an active node to guess the IHCs of passive nodes in the same cell, it first runs Algorithm 15 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. 6.2(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. 6.2(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 .

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, $L_a = \emptyset$, $L_p = \emptyset$, and $L_c = \emptyset$. 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 the above step is finished, 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, then each active node needs to wait until every active node completes the channel hopping.

Success Probability of Self-Organization. In this part, 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. 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 k th channel of the $\Delta M + \Delta' M + 1$ channels.

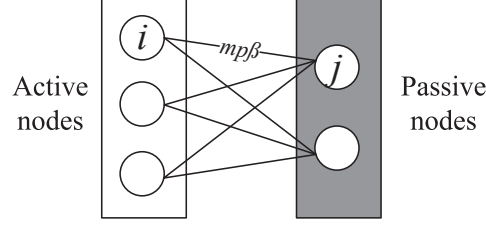


Figure 6.3: The probability of connection as a bipartite graph

Let $X_k = 1$ or 0 denote if the k th 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.

Lemma 6.3. [92] *Let Z_{jk} denote the event that passive node j selects the k th channel in the range $[IH_j - \Delta M, IH_j + \Delta' M]$ as its home channel using Algorithm 17. Then the probability of Z_{jk} , denoted as β , is given as*

$$\beta = \Pr(Z_{jk}) = \frac{1}{m} (1 - (1 - p)^m). \quad (6.1-1)$$

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 6.4. *The P_s is lower bounded as follows,*

$$P_s \geq 1 - (1 - m\beta p)^{\min(\tilde{N}, \bar{N})}. \quad (6.1-2)$$

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

$$\sum_{k=1}^m \beta p = m\beta p. \quad (6.1-3)$$

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. (6.1-3), the connection probability of every link is $m\beta p$, since (6.1-3) 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

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

□

Estimation of Channel Hopping Range. Based on Theorem 6.4, 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. 6.4 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).

6.1.4 Backbone Nodes Selection

The classical clustering approach [50] 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

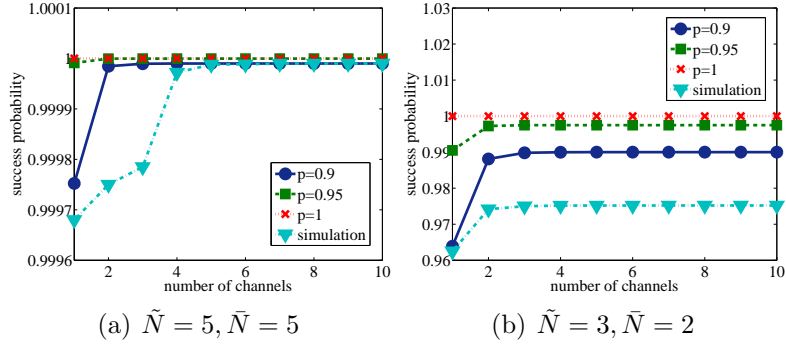


Figure 6.4: The self-organization success probability

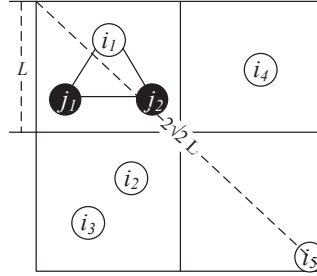


Figure 6.5: Example of backbone nodes selection.

connected 1-hop neighbors are marked as covered; repeat (2) and (3) on all uncovered nodes (if any).

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, 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. 6.5, 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 6.5. *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. \square

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 [50], 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 [50]. The priority among nodes still depends on their ID. The remaining problem is how the cluster heads exchange information with each other in order to run the above marking process and pruning rule.

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

Algorithm 16 h sends requests to cluster heads in an adjacent cell c'_k

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

reach the adjacent eight cells, which covers all the cluster heads within r_2 around it. For example, node i_5 in Fig. 6.5 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 16.

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. 6.5, 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 $IH_{i_2} \notin M_{i_1}$, 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.

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 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 6.6. *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 [50], 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 [50] 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 the approach in [50], which ensures the connectivity and the coverage using the same broadcast process in [50]. □

6.1.5 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.

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).

THC Selection. We adopt the approach in [40] for the nodes to choose THCs and guess the THCs of others, which has minimal control overhead and is highly effective. Algorithm 17 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 17 is similar to Algorithm ???. The difference 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 [40].

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 17 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 [40]. This approach does not require any exchange of control messages, which significantly

Algorithm 17 $Pro2(i, M_i)$, 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 k th channel in \mathcal{Q} }
 - 6: $\mathcal{Q} = \mathcal{Q} \setminus \{q\}$ {Remove q from \mathcal{Q} }
 - 7: **until** $q \in M_i$
 - 8: Return q {Selected THC}
-

streamlines the communication and reduces overhead.

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 develop 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 .

Variation 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 $[g_1, g_2, \dots, g_n]$ 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 $[g_1, g_2, \dots, g_n]$ 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 $[g_1, g_2, \dots, g_n]$ 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.

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.

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 6.7. 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. 6.1, the area route from source S to destination D is $A_1 \rightarrow$

$A_2 \rightarrow \dots \rightarrow A_k$. 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.

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 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 $[g_1, g(2), \dots, g_{N'}]$; 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.

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 $\{A_1, A_2, \dots, A_k\}$ 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.

6.1.6 Simulations

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 assume that the bandwidth of each channel is the same. The active state and passive state each lasts 1 time slot. Each node switches among the two states independently. We compare the throughput and delay with the approach in [40], which does not have a virtual backbone construction.

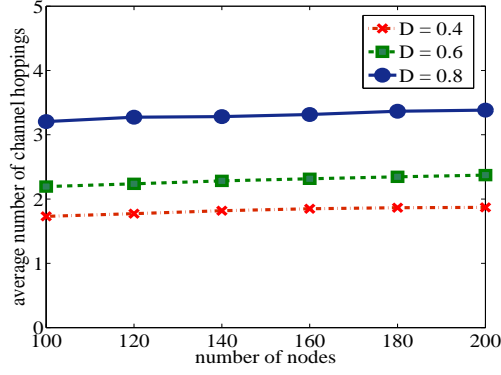


Figure 6.6: Number of hoppings VS nodes.

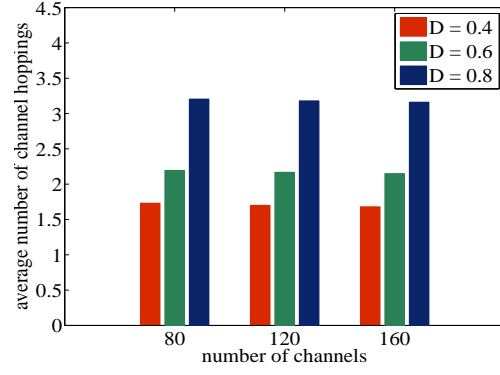


Figure 6.7: Number of hoppings VS channels.

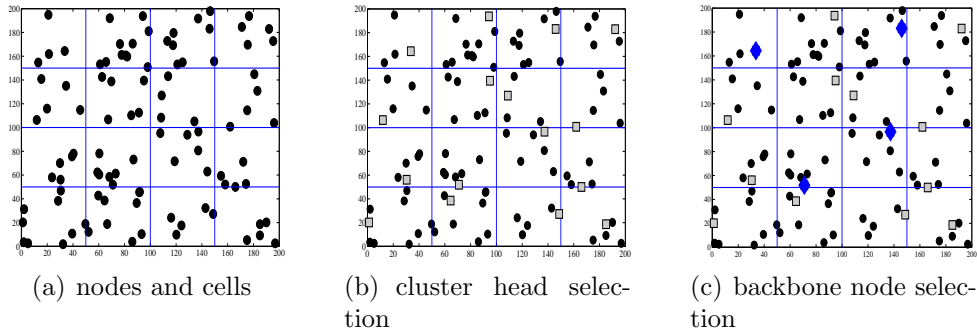


Figure 6.8: Process of a backbone construction.

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. 6.6 and 6.7. We compare the average number of channel hoppings needed to achieve three different channel busy time ratios, i.e., $\frac{u}{u+v} = 0.4, 0.6, \text{ and } 0.8$. The results show that the number of channel hoppings needed for all three are less than 5, which verifies our previous analysis. Fig. 6.6 shows that when the number of nodes increases, the number of channel hoppings increases slowly. Fig. 6.7 shows that the increase in the channel availabilities 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 node selection using our model. The process is shown in Fig. 6.8. Fig. 6.8(a) shows the nodes distribution in the network with

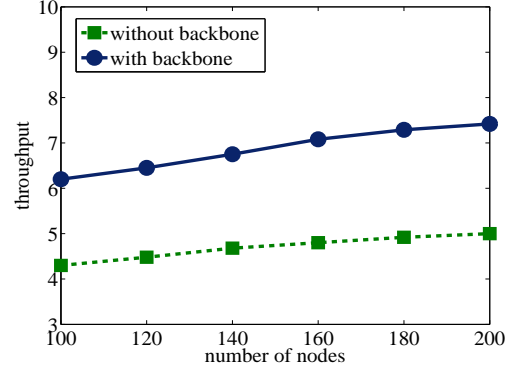
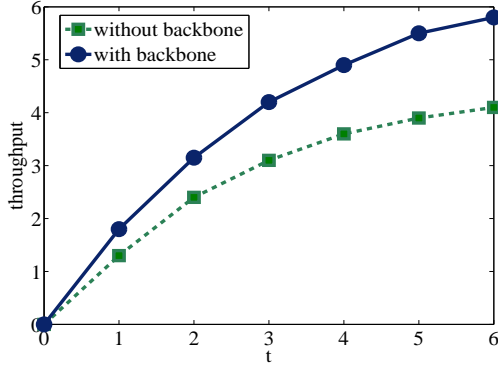


Figure 6.9: Throughput over time. Figure 6.10: Throughput *VS* nodes.

cell divisions. In Fig. 6.8(b) the square nodes are the selected cluster heads from each cell. Fig. 6.8(c) shows the selected backbone nodes (the diamond ones).

Moreover, we implement the algorithm in [40], 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 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. 6.9, 6.10, and 6.11. The throughput is the average throughput for all the active sessions in the network. In Fig. 6.9, 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. 6.10, the results show that the throughputs of both approaches increase slowly when the number of nodes increases. In Fig. 6.11, the throughputs of both approaches increase while the number of channels increases. Overall, the throughput with virtual backbone structure is much larger than the one without virtual backbone structure.

At last, we compare the packet delay between the two approaches. The results are shown in Figs. 6.12, 6.13, and 6.14. In Fig. 6.12, we vary the number of active

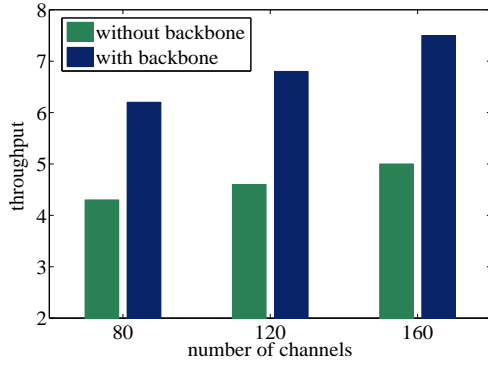


Figure 6.11: Throughput VS channels.

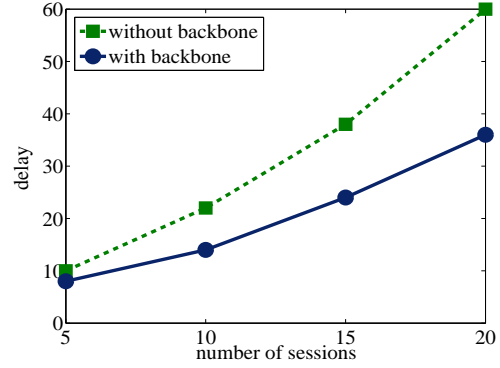


Figure 6.12: Delay VS session number.

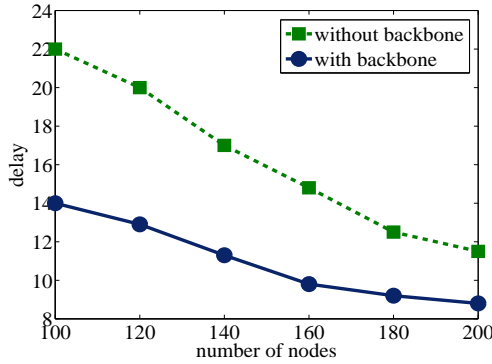


Figure 6.13: Delay VS number of nodes

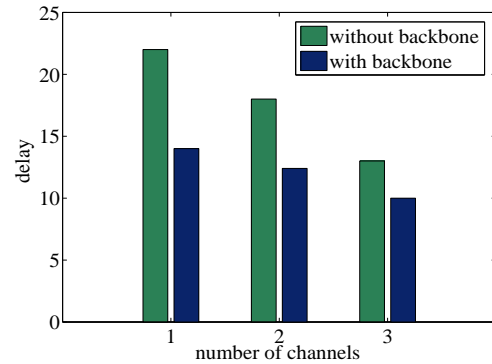


Figure 6.14: Delay VS number of channels.

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. 6.13, 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. 6.14, 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.

6.2 Chapter Conclusion

In this chapter, we focus on the infrastructure design in CRNs to improve the performance of data transmission. We present a comprehensive approach on self-organization, virtual backbone construction, and end-to-end data transmission for CRNs, without relying on a 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.

CHAPTER 7

CONCLUSION

This dissertation presents our in-depth research works on the spectrum management and cross-layer protocol designs in CRNs. The highly dynamic channel availabilities and unpredictable PU activities pose extra challenges to works in CRNs. We propose our models and algorithms to solve the challenges in CRNs, which improve the performances from different aspects. We summarize our contributions and point out some future research topic, that are very promising.

7.1 Summary of Contributions

CRNs are different from traditional wireless networks. There are two types of users in CRNs, which are PUs and SUs. PUs have higher priorities on channel access, and SUs need to quit channel usage once the nearby PUs become active. Moreover, due to security concerns, the active patterns of PUs are unpredictable to SUs. The channel dynamics make the spectrum sharing, session construction and maintenance difficult.

First of all, this dissertation start with improving the performance of spectrum sensing, which is the key phase to protect PUs. We propose a pre-phase improvement for sensing. We also give a hitchking scheme, which makes use of the core structures to assist spectrum sensing. Through our approaches, the sensing accuracy is improved while the time cost for spectrum sensing is reduced.

Secondly, we study the channel assignment, which is an important part of spectrum management. We propose distributed channel assignment algorithms, which is

effective and conflict-free. In addition, given a specific routing protocol, we make use of the control messages to collect information. The collected information is used to facilitate a more effective channel assignment scheme.

Next, we study the routing protocols in CRNs. Different from routings in traditional wireless networks, the special challenges of CRNs motivate us to propose a novel routing protocols with the assistance of boundary nodes. We take the angle dimensions into accounts, and select routes that pass less PU areas and are more reliable through the information of boundary nodes. In addition, we study the forwarding node set selections for routing use, and propose a multi-layer algorithm.

Last but not least, we consider the infrastructure design for CRNs, which could be very helpful for multiple aspects in CRNs, e.g., channel assignment, link construction, and routing protocols. We construct the virtual backbone without a common control channel. Nodes are self-organized into cells and backbone nodes are distributedly selected from them. We also give the link construction algorithm, and area routing for end-to-end data transmissions.

7.2 Future Research

Since the spectrum usage is increasing rapidly nowadays and the previous static spectrum allocation scheme can barely meet today's requirements, how to apply the cognitive radio technology is becoming more important and inevitable. In the future research, I plan to continue working on improving the reliability and practicability of CRNs.

The first direction is the application of geolocation databases [93–98]. Instead of purely relying on spectrum sensing [99], the geolocation databases can be used to store the static information about channel availabilities and spectrum maps. For example, IEEE 802.11af [100] is about the standard for spectrum sharing in TV white spaces based on spectrum databases. Due to environmental differences and security

issues, the integration of database information and spectrum sensing results will be a reasonable and promising way for future use. It reduces the cost of spectrum sensing, and improves the accuracy of the generated spectrum map based on the integration results.

The second direction lies in the performance evaluation of the protocols in CRNs [101–103]. The current evaluation results are mainly based on simulations, and experimental results on a small scale of USRP testbeds. A large scale of collected channel usage data samples, which are over a long time duration and a large geography area, will be helpful for practical performance evaluations. Also, the spectrum management approaches and cross-layer protocols can be evaluated in a long term with more practical channel dynamics. The implementation of the large-scale testbed has huge potentials and are necessary in order to make CRNs practical in real industrial environment.

BIBLIOGRAPHY

- [1] FCC, “Notice of proposed rule making and order,” ET Docket No 03-222, 2003.
- [2] 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*, vol. 50, pp. 2127–2159, 2006.
- [3] T. Yucek and H. Arslan, “A survey of spectrum sensing algorithms for cognitive radio applications,” *IEEE Communications Surveys Tutorials*, 2009.
- [4] G. K. Audhya, K. Sinha, S. C. Ghosh, and B. P. Sinha, “A survey on the channel assignment problem in wireless networks,” *Wireless Communications and Mobile Computing*, 2010.
- [5] M. Cesana, F. Cuomo, and E. Ekici, “Routing in cognitive radio networks: Challenges and solutions,” *Ad Hoc Networks*, 2010.
- [6] A. Cacciapuoti, I. Akyildiz, and L. Paura, “Correlation-aware user selection for cooperative spectrum sensing in cognitive radio ad hoc networks,” *IEEE Journal on Selected Areas in Communications*, 2012.
- [7] J. Ma, G. Zhao, and Y. Li, “Soft combination and detection for cooperative spectrum sensing in cognitive radio networks,” *IEEE Transactions on Wireless Communications*, 2008.
- [8] R. Fan and H. Jiang, “Optimal multi-channel cooperative sensing in cognitive radio networks,” *IEEE Transactions on Wireless Communications*, 2010.
- [9] E. Peh, Y.-C. Liang, Y. L. Guan, and Y. Zeng, “Optimization of cooperative sensing in cognitive radio networks: A sensing-throughput tradeoff view,” *IEEE Transactions on Vehicular Technology*, 2009.
- [10] H. Li, “Customer reviews in spectrum: Recommendation system in cognitive radio networks,” in *Proc. of IEEE DySPAN*, 2010.
- [11] H. Li, “A recommendation system in cognitive radio networks with random data traffic,” *IEEE Transactions on Vehicular Technology*, 2011.
- [12] W. Hale, “Frequency assignment: theory and application,” in *Proc. of the IEEE*, vol. 68, pp. 1497–1514, 1980.

- [13] D. Marx, “NP-completeness of list coloring and precoloring extension on the edges of planar graphs,” *Journal of Graph Theory*, vol. 49, pp. 313–324, 2004.
- [14] T. Shu and M. Krunz, “Joint power/rate control and channel assignment in cognitive radio networks: A multi-level spectrum opportunity perspective,” *University of Arizona, Department of ECE, Tech. Rep.*, 2008.
- [15] Y. Shi, Y. T. Hou, S. Kompella, and H. D. Sherali, “Maximizing capacity in multi-hop cognitive radio networks under the SINR model,” *IEEE Transactions on Mobile Computing*, vol. 99, no. PrePrints, 2010.
- [16] A. Panconesi and R. Rizzi, “Some simple distributed algorithms for sparse networks,” *Distributed Computing*, vol. 14(2), pp. 97–100, 2001.
- [17] A. V. Goldberg, S. A. Plotkin, and G. E. Shannon, “Parallel symmetry-breaking in sparse graphs,” in *SIAM J. Disc. Math*, pp. 315–324, 1987.
- [18] G. De Marco and A. Pelc, “Fast distributed graph coloring with $o(\delta)$ colors,” in *Proc. of ACM/SIAM SODA*, SODA ’01, pp. 630–635, 2001.
- [19] W. Wang and X. Liu, “List-coloring based channel allocation for open-spectrum wireless networks,” in *Proc. of IEEE VTC*, 2005.
- [20] C. Xin, L. Ma, and C.-C. Shen, “A path-centric channel assignment framework for cognitive radio wireless networks,” *Mob. Netw. Appl.*, vol. 13, no. 5, 2008.
- [21] X. Zhou, L. Lin, J. Wang, and X. Zhang, “Cross-layer routing design in cognitive radio networks by colored multigraph model,” *Wireless Personal Communications*, 2009.
- [22] Y. Liu, L. Cai, and X. Shen, “Spectrum-aware opportunistic routing in multi-hop cognitive radio networks,” *IEEE Journal on Selected Areas in Communications*, 2012.
- [23] K. R. Chowdhury and I. F. Akyildiz, “CRP: A routing protocol for cognitive radio ad hoc networks,” *IEEE Journal on Selected Areas in Communications*, 2011.
- [24] A. Raniwala, K. Gopalan, and T. Chiueh, “Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks,” *ACM Mobile Computing and Communications*, 2004.
- [25] J. So and N. H. Vaidya, “Routing and channel assignment in multi-channel multi-hop wireless networks with single-nic devices,” *Technical Report*, 2004.
- [26] H. Wu, F. Yang, K. Tan, J. Chen, Q. Zhang, and Z. Zhang, “Distributed channel assignment and routing in multi-radio multi-channel multi-hop wireless networks,” *Technical Report*, 2006.

- [27] 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.
- [28] G. Zhao, J. Ma, G. Y. Li, T. Wu, Y. Kwon, A. Soong, and C. Yang, "Spatial spectrum holes for cognitive radio with relay-assisted directional transmission," *IEEE Transactions on Wireless Communications*, 2009.
- [29] D. Wilcox, E. Tsakalaki, A. Kortun, T. Ratnarajah, C. Papadias, and M. Sellathurai, "On spatial domain cognitive radio using single-radio parasitic antenna arrays," *IEEE Journal on Selected Areas in Communications*, 2013.
- [30] W. Guo and X. Huang, "Multicast communications in cognitive radio networks using directional antennas," *Wireless Communications and Mobile Computing*, 2012.
- [31] W. Feng, J. Cao, C. Zhang, and C. Liu, "Joint optimization of spectrum handoff scheduling and routing in multi-hop multi-radio cognitive networks," in *Proc. of IEEE ICDCS*, 2009.
- [32] A. Abbagnale and F. Cuomo, "Gymkhana: A connectivity-based routing scheme for cognitive radio ad hoc networks," in *Proc. of IEEE INFOCOM*, 2010.
- [33] I. Filippini, E. Ekici, and M. Cesana, "Minimum maintenance cost routing in cognitive radio networks," in *Proc. of IEEE MASS*, 2009.
- [34] H. Ma, L. Zheng, X. Ma, and Y. Luo, "Spectrum aware routing for multi-hop cognitive radio networks with a single transceiver," in *Proc. of IEEE Crown-Com*, 2008.
- [35] G. Cheng, W. Liu, Y. Li, and W. Cheng, "Spectrum aware on-demand routing in cognitive radio networks," in *Proc. of IEEE DySPAN*, 2007.
- [36] Z. Yang, G. Cheng, W. Liu, W. Yuan, and W. Cheng, "Local coordination based routing and spectrum assignment in multi-hop cognitive radio networks," *Mobile Networks and Applications*, 2008.
- [37] S. Liu, L. Lazos, and M. Krunz, "Cluster-based control channel allocation in opportunistic cognitive radio networks," *IEEE Transactions on Mobile Computing*, 2012.
- [38] M. Ozger and O. Akan, "Event-driven spectrum-aware clustering in cognitive radio sensor networks," in *Proc. of IEEE Infocom*, 2013.
- [39] L. DaSilva and I. Guerreiro, "Sequence-based rendezvous for dynamic spectrum access," in *Proc. of IEEE DySpan*, 2008.

- [40] 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.
- [41] 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.
- [42] 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.
- [43] 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.
- [44] C.-M. Chao and H.-Y. Fu, "Providing complete rendezvous guarantee for cognitive radio networks by quorum systems and latin squares," in *Proc. of IEEE WCNC*, 2013.
- [45] W. Yuan, H. Leung, W. Cheng, S. Cheng, and B. Chen, "Participation in repeated cooperative spectrum sensing: A game-theoretic perspective," *IEEE Transactions on Wireless Communications*, 2012.
- [46] Y. Zhao, M. Song, C. Xin, and M. Wadhwa, "Spectrum sensing based on three-state model to accomplish all-level fairness for co-existing multiple cognitive radio networks," in *Proc. of IEEE INFOCOM*, 2012.
- [47] J. Laska, B. Bradley, T. Rondeau, K. Nolan, and B. Vigoda, "Compressive sensing for dynamic spectrum access networks: techniques and tradeoffs," in *Proc. of IEEE DySPAN*.
- [48] J. Chiang and Y. Hu, "Did you also hear that? spectrum sensing using hermitian-inner-product," in *Proc. of IEEE Infocom*, 2013.
- [49] Q. Liu and Y. Cui, "Scheduling of sequential periodic sensing for cognitive radios," in *Proc. of IEEE Infocom*, 2013.
- [50] J. Wu and F. Dai, "Virtual backbone construction in manets using adjustable transmission ranges," *IEEE Transactions on Mobile Computing*.
- [51] Y. Yuan, P. Bahl, R. Chandra, T. Moscibroda, and Y. Wu, "Allocating dynamic time-spectrum blocks in cognitive radio networks," in *Proc. of ACM Mobihoc*, 2007.
- [52] A. T. Hoang and Y.-C. Liang, "Maximizing spectrum utilization of cognitive radio networks using channel allocation and power control," in *Proc. of IEEE VTC*, 2006.

- [53] A. T. Hoang and Y. C. Liang, "A two-phase channel and power allocating schemes for cognitive radio networks," in *Proc. of IEEE PIMRC*, 2006.
- [54] Y. Ding and L. Xiao, "Routing and spectrum allocation for video on-demand streaming in cognitive wireless mesh networks," in *Proc. of IEEE MASS*, 2010.
- [55] E. Tragos, A. Fragkiadakis, I. Askoxylakis, and V. Siris, "The impact of interference on the performance of a multi-path metropolitan wireless mesh network," in *Proc. of the 16th IEEE Symposium on Computers and Communications (ISCC' 11)*, 2011.
- [56] J. A. Bondy and U. Murthy, *Graph Theory with Applications*. Elsevier Ltd, 1976.
- [57] S. Yang, J. Wu, and F. Dai, "Efficient directional network backbone construction in mobile ad hoc networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, December 2008.
- [58] L. Yang, L. Cao, and H. Zheng, "Physical interference driven dynamic spectrum management," in *Proc. of IEEE DySPAN*, 2008.
- [59] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," *Kluwer Academic*, 1996.
- [60] O. Goussevskaia, R. Wattenhofer, M. M. Halldorsson, and E. Welzl, "Capacity of arbitrary wireless networks," in *Proc. of IEEE Infocom*, 2009.
- [61] M. Timmers, S. Pollin, A. Dejonghe, and A. Bahai, "Accumulative interference modeling for distributed cognitive radio networks," *Journal of Communications*, 2009.
- [62] G. Brar, D. M. Blough, and P. Santi, "Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks," in *Proc. of ACM MobiCom*, 2006.
- [63] L. Cao, L. Yang, X. Zhou, Z. Zhang, and H. Zheng, "Optimus: SINR-driven spectrum distribution via constraint transformation," in *Proc. of IEEE DySPAN*, 2010.
- [64] D. P. Bertsekas, *Linear Network Optimization*. The MIT Press, 1991.
- [65] B. Yang, J. Geunes, and W. J. O'Brien, "Resource-constrained project scheduling: Past work and new directions," *Technical Report*, 2001.
- [66] F. Dai and J. Wu, "Proactive route maintenance in wireless ad hoc networks," *Proc. of IEEE ICC*, 2005.
- [67] R. Vilzmann and C. Bettstetter, "A survey on mac protocols for ad hoc networks with directional antennas," in *EUNICE 2005: Networks and Applications Towards a Ubiquitously Connected World*, 2006.

- [68] F. Dai and J. Wu, "Efficient broadcasting in ad hoc wireless networks using directional antennas," *IEEE Transactions on Parallel and Distributed Systems*, 2006.
- [69] M. Abdel-Rahman, H. Rahbari, M. Krunz, and P. Nain, "Fast and secure rendezvous protocols for mitigating control channel dos attacks," in *Proceedings of IEEE INFOCOM*, 2013.
- [70] I. F. Akyildiz, W.-Y. Lee, M. Vuran, and S. Mohanty, "A survey on spectrum management in cognitive radio networks," *IEEE Communications Magazine*, 2008.
- [71] W.-Y. Lee and I. F. Akyildiz, "Spectrum-aware mobility management in cognitive radio cellular networks," *IEEE Transactions on Mobile Computing*, 2012.
- [72] M. Lu, F. Li, and J. Wu, "Efficient opportunistic routing in utility-based ad hoc networks," *IEEE Transactions on Reliability*, 2009.
- [73] Y. Zhang, N. Meratnia, and P. Havinga, "Outlier detection techniques for wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, 2010.
- [74] J. T. Chiang, D. Kim, and Y.-C. Hu, "Jim-beam: Using spatial randomness to build jamming-resilient wireless flooding networks," in *Proc. of ACM Mobihoc*, 2012.
- [75] X. Zhou, R. Ganti, and J. Andrews, "Secure wireless network connectivity with multi-antenna transmission," *IEEE Transactions on Wireless Communications*, 2011.
- [76] S. Biswas and R. Morris, "Exor: Opportunistic multi-hop routing for wireless networks," in *Proc. of ACM Sigcomm*, 2005.
- [77] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," in *Proc. of ACM Sigcomm*, 2007.
- [78] E. Rozner, J. Seshadri, Y. Mehta, and L. Qiu, "Soar: Simple opportunistic adaptive routing protocol for wireless mesh networks," *IEEE Transactions on Mobile Computing*, 2009.
- [79] U. Lee, P. Wang, Y. Noh, F. Vieira, M. Gerla, and J.-H. Cui, "Pressure routing for underwater sensor networks," in *Proc. of IEEE Infocom*, 2010.
- [80] Z. Wang, Y. Chen, and C. Li, "Corman: A novel cooperative opportunistic routing scheme in mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, 2012.
- [81] N. Zhi, L. Jing, L. Botong, L. Hanchun, and X. Youyun, "A relay node selection technique for opportunistic routing in mobile ad hoc networks," in *15th Asia-Pacific Conference on Communications*, 2009.

- [82] F. Baccelli, B. Blaszczyszyn, E. Ermel, and P. Muhlethaler, “An optimized relay self selection technique for opportunistic routing in mobile ad hoc networks,” in *European Wireless Conference (EW)*, 2008.
- [83] M. Stojanovic, “On the relationship between capacity and distance in an underwater acoustic communication channel,” in *Proc. of ACM WUWNet*, 2006.
- [84] T. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2nd ed., 2001.
- [85] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris, “A high-throughput path metric for multi-hop wireless routing,” in *Proc. of ACM/IEEE MobiCom*, 2003.
- [86] R. Draves, J. Padhye, and B. Zill, “Comparison of routing metrics for static multi-hop wireless networks,” in *Proc. of ACM Sigcomm*, 2004.
- [87] V. Lozin and M. Milanic, “A polynomial algorithm to find an independent set of maximum weight in a fork-free graph,” *Journal of Discrete Algorithms*, 2008.
- [88] B. Liang and Z. Haas, “Virtual backbone generation and maintenance in ad hoc network mobility management,” in *Proc. of IEEE Infocom*, 2000.
- [89] K. Alzoubi, P. Wan, and O. Frieder, “New distributed algorithm for connected dominating set in wireless ad hoc networks,” in *Proc. of Hawaii Int’l Conf. System Sciences*, 2002.
- [90] K. Alzoubi, P. Wan, and O. Frieder, “Message-optimal connected dominating sets in mobile ad hoc networks,” in *Proc. of ACM MobiHoc*, 2002.
- [91] 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.
- [92] 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.
- [93] X. Chen and J. Huang, “Game theoretic analysis of distributed spectrum sharing with database,” in *Proc. of IEEE ICDCS*, 2012.
- [94] J. Ojaniemi, J. Poikonen, and R. Wichman, “Effect of geolocation database update algorithms to the use of TV white spaces,” in *Proc. of Crowncom*, 2012.
- [95] A. Achtzehn, J. Riihijarvi, and P. Mahonen, “Improving accuracy for TVWS geolocation databases: Results from measurement-driven estimation approaches,” in *Proc. of IEEE DySPAN*, 2014.
- [96] B. Gao, J.-M. Park, and Y. Yang, “Supporting mobile users in database-driven opportunistic spectrum access,” in *Proc. of ACM MobiHoc*, 2014.

- [97] K. Zeng, S. K. Ramesh, and Y. Yang, "Location robustness in database-driven white spaces network," in *Proc. of IEEE DySPAN*, 2014.
- [98] Z. Gao, H. Zhu, Y. Liu, M. Li, and Z. Cao, "Location privacy in database-driven cognitive radio networks: Attacks and countermeasures," in *Proc. of IEEE INFOCOM*, 2013.
- [99] R. Murty, R. Chandra, T. Moscibroda, and P. Bahl, "SenseLess: A database-driven white spaces network," *IEEE Transactions on Mobile Computing*, 2012.
- [100] A. Flores, R. Guerra, E. Knightly, P. Ecclesine, and S. Pandey, "IEEE 802.11af: A standard for TV white space spectrum sharing," *IEEE Communications Magazine*, 2013.
- [101] A. Asal, A. Mamdouh, A. Salama, M. Elgendy, M. Mokhtar, M. Elsayed, and M. Youssef, "Crescent: A modular cost-efficient open-access testbed for cognitive radio networks routing protocols," in *Proc. of ACM Mobicom*, 2013.
- [102] S. Soltani, Y. Sagduyu, J. Li, J. Feldman, and J. Matyjas, "Demonstration of plug-and-play cognitive radio network emulation testbed," in *Proc. of IEEE DySPAN*, 2014.
- [103] V. Passas, K. Chounos, S. Keranidis, W. Liu, L. Hollevoet, T. Korakis, I. Koutsopoulos, I. Moerman, and L. Tassiulas, "Online evaluation of sensing characteristics for radio platforms in the crew federated testbed," in *Proc. of ACM Mobicom*, 2013.

PUBLICATION LIST

Journal Papers

1. **Ying Dai**, Jie Wu, and Chunsheng Xin, “Efficient Virtual Backbone Construction without a Common Control Channel in Cognitive Radio Networks,” accepted to appear in *IEEE Transactions on Parallel and Distributed System*.
2. **Ying Dai**, Jie Wu, and Yaxiong Zhao, “Boundary Helps: Reliable Route Selection With Directional Antennas in Cognitive Radio Networks,” accepted to appear in *IEEE Transactions on Vehicular Technology*.
3. **Ying Dai**, Jie Wu, and Andrew Daniels, “Effective Channel Assignment Based on Dynamic Source Routing in Cognitive Radio Networks,” accepted to appear in *Ad Hoc & Sensor Wireless Networks*.
4. **Ying Dai** and Jie Wu, “Cooperation Scheme For Distributed Spectrum Sensing In Cognitive Radio Networks,” accepted to appear in *Mobile Communications and Applications*.
5. **Ying Dai** and Jie Wu, “Pre-phase Improvement For Distributed Spectrum Sensing in Cognitive Radio Networks,” accepted to appear in *Mobile Communications and Applications*.
6. Jie Wu, **Ying Dai**, and Yanchao Zhao, “Effective Channel Assignments in

Cognitive Radio Networks,” *Computer Communications*, Vol. 36, No. 4, 411-420, 2013.

Conference Papers

7. **Ying Dai** and Jie Wu, “Hitchhiking For Sensing: Distributed Spectrum Sensing Assisted By Core and Cluster Structures in Cognitive Radio Networks,” *Proceedings of the 23rd International Conference on Computer Communications and Networks (ICCCN)*, August 4-7, 2014.
8. **Ying Dai**, Jie Wu, and Andrew Daniels, “Channel Dynamics Matter: Forwarding Node Set Selection in Cognitive Radio Networks,” *Proceedings of the 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks (Short Paper) (WoWMoM)*, June 16-19, 2014.
9. Pouya Ostovari, Jie Wu, and **Ying Dai**, “Priority-Based Broadcasting of Sensitive Data in Error-Prone Wireless Networks,” *Proceedings of the 2nd International Symposium on Resilient Communication Systems (ISRCS)*, August 19-21, 2014.
10. **Ying Dai**, Jie Wu, and Chunsheng Xin, “Virtual Backbone Construction for Cognitive Radio Networks without Common Control Channel,” *Proceedings of the 32nd IEEE International Conference on Computer Communications (INFOCOM)*, April 14-19, 2013.
11. **Ying Dai** and Jie Wu, “Boundary Helps: Efficient Routing Protocol using Directional Antennas in Cognitive Radio Networks,” *Proceedings of the 10th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, October 14-16, 2013.
12. **Ying Dai** and Jie Wu, “Sense in Order: Channel Selection for Sensing in

- Cognitive Radio Networks,” *Proceedings of the 8th International Conference on Cognitive Radio Oriented Wireless Networks* (CROWNCOM), July 8-10, 2013.
13. **Ying Dai** and Jie Wu, “Whether and When to Share: Spectrum Sensing as An Evolutionary Game,” *Proceedings of the 8th International Conference on Cognitive Radio Oriented Wireless Networks* (CROWNCOM), July 8-10, 2013.
 14. **Ying Dai** and Jie Wu, “Distributed Rerouting For Multiple Sessions in Cognitive Radio Networks,” *Proceedings of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications* (Trust-Com), June 25-27, 2012.
 15. **Ying Dai** and Jie Wu, “Efficient Channel Assignment Under Dynamic Source Routing in Cognitive Radio Networks,” *Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems* (MASS), October 17-22, 2011.
 16. Jie Wu, **Ying Dai**, and Yanchao Zhao, “Local Channel Assignments in Cognitive Radio Networks,” *Proceedings of the 20th International Conference on Computer Communications and Networks* (ICCCN), July 31 - August 4, 2011.
 17. **Ying Dai**, Panlong Yang, Guihai Chen, and Jie Wu, “CFP: Integration of Fountain Codes and Optimal Probabilistic Forwarding in DTNs,” *Proceedings of the IEEE Global Telecommunications Conference* (GLOBECOM), December 6-10, 2010.