SECURING WIRELESS BROADCAST COMMUNICATION AGAINST INTERNAL ATTACKS

by

Sisi Liu

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2 0 1 1

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Final Examination Committee, we certify that we have read the dissertation prepared by Sisi Liu
entitled Securing Wireless Broadcast Communication against Internal Attacks
and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

Dr. Loukas Lazos — Date: 9 Dec 2011

Dr. Roman Lysecky — Date: 9 Dec 2011

Date: 9 Dec 2011

Date: 9 Dec 2011

Date: 9 Dec 2011

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Dissertation Director: Dr. Marwan Krunz — Date: 9 Dec 2011

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: _____

Sisi Liu

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my advisors, Professor Marwan Krunz and Professor Loukas Lazos, for all the help, advice, supports and encouragements they gave to me during the past four years. Because of them, my graduate experience has been the one that I will cherish forever. I am fortunate to have them as my advisors who taught me how to think and research. They have enlightened me through their professional knowledge about where to explore and what is necessary to get there. This dissertation would have been impossible without their guidance.

I would like to thank Professor Roman Lysecky for all the valuable suggestions and discussions that make this dissertation better.

I would like to thank all my labmates in the wireless networking group for their help and friendship, including Diep Nguyen, Junseok Kim, Hanif Rahbari, Mohammad Jamal Abdel Rahman, Mohammed Alfowzan and Harish Kumar Shankar. I wish to give special thanks to Shu Tao who have graduated from this group for his valuable discussions and patient help. I would also thank Tami Whelan for handling all the paperwork and giving various forms of support during my graduate study. I would like to thank my parents and my husband Meng Zeng for supporting and encouraging me throughout my Ph.D. study. Their love and care give me strength to overcome difficulties throughout this endeavor.

DEDICATION

To my parents and Meng Zeng.

TABLE OF CONTENTS

TABLE OF CONTENTS – *Continued*

TABLE OF CONTENTS – *Continued*

LIST OF FIGURES

LIST OF FIGURES – *Continued*

LIST OF FIGURES – *Continued*

LIST OF FIGURES – *Continued*

LIST OF TABLES

ABSTRACT

Coordination of network functions in wireless networks requires frequent exchange of control messages among participating nodes. Typically, such messages are transmitted over a universally known communication channel referred to as the control channel. The location of the control channel, determined by its frequency band, time slot, or spreading code, is known a priori to all nodes participating in the network. Due to its critical role, this channel can become a prime target of Denial-of-Service (DoS) attacks. In this dissertation, we propose several schemes to preventing control-channel DoS attacks, manifested in the form of jamming. We propose a control channel allocation scheme for dynamic-spectrum networks (i.e., cognitive networks). In our work, we assume a sophisticated adversary, who has knowledge of the protocol specifics and of the cryptographic quantities used to secure network operations. Such an adversarial model cannot be addressed by classic anti-jamming techniques that rely on shared secrets (e.g., spread spectrum). New security metrics are proposed to quantify the ability of the adversary to deny access to the control channel. We introduce a randomized distributed scheme that allows nodes to establish and maintain the control channel in the presence of the jammer. This scheme uniquely identifies the set of compromised nodes, both when nodes are acting independently and when they are colluding. To protect broadcast communication performed on control channel, we also propose a time-delayed broadcast scheme (TDBS), which implements the broadcast operation as a series of unicast transmissions, distributed in frequency and time. Finally, we address the problem of dynamically assigning the control channel in CRNs based on time- and space-varying spectrum opportunities. We propose a cluster-based architecture that allocates different channels for control at various clusters in the network. The clustering problem is formulated as a bipartite graph problem, for which we develop

a class of algorithms to implement. Extensive simulations are conducted to verify the validity of the proposed mechanisms.

CHAPTER 1

Introduction

1.1 Background

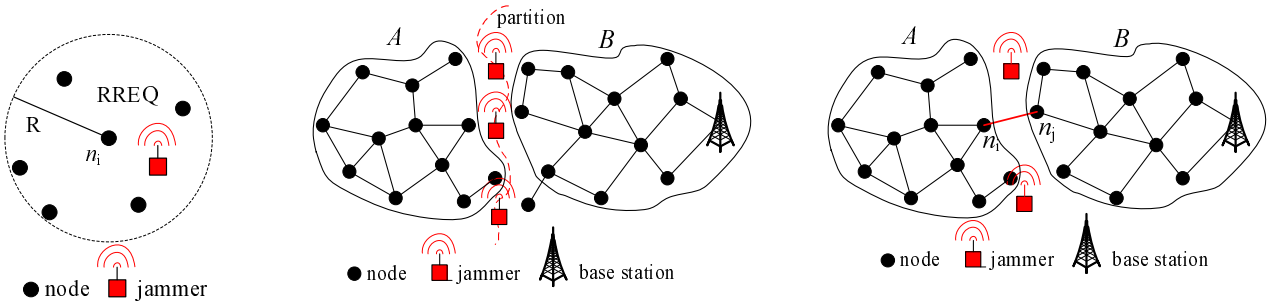
1.1.1 Classic jamming approach

The open nature of wireless medium makes the wireless network vulnerable to intentional interference attacks, typically referred to as jamming. In the simplest form of jamming, the adversary interferes with the signal reception by transmitting a continuous jamming waveform [73] or several short jamming pulses [59]. By doing this, the adversary can block the wireless medium and prevents other wireless devices from communicating. Jamming attack belongs to Denial-of-Service (DOS) attack. Traditional Denial-of-Service attacks are attacks against availability and attempting to prevent legitimate users from accessing the network [12]. In the context of wireless domain, the adversary with jamming device is more powerful in denying access to the wireless medium and degrading network performance.

Typically, jamming attacks have been analyzed and addressed as a physical-layer vulnerability. There are many different jamming strategies that a jammer can perform in order to interfere with wireless communications.

Constant jamming and random jamming: Based on the interval between the state of jamming and not jamming, jamming can be classified into constant jamming and random jamming. Constant jammer continually emits a radio signal without following any specific protocol. Specifically, the constant jammer does not wait for the channel to become idle before transmitting [90]. Instead of continuously sending out radio signal, random jamming alternates between sleeping and jamming.

Active jamming and reactive jamming: Based on the status of the channel when jamming attack happens, there are active jamming and reactive jamming. The



(a) (b) (c)

Figure 1.1: (a) The adversary jams route requests (RREQ) broadcasted by node n_i on the control channel, (b) the adversary partitions the network into two components A and B, by deploying multiple jamming devices, so nodes in A cannot alert their base station regarding the jamming attack, (c) the adversary forces all traffic from A to B to pass through the link n_i, n_j.

difference between the two lies in that a jammer with reactive jamming strategy stays quiet when the channel is idle, but starts transmitting a radio signal as soon as it senses activity on the channel, while the active jammer jams no matter the channel is idle or not.

Selective jamming: According to the content of communication that the jamming targets on, jamming attack will be classified into selective jamming and non-selective jamming. Selective jamming selectively and adaptively chooses critical network functions as jamming target. Selective jamming can be further categorized into channel-selective jamming and data-selective jamming [46]. Channel-selective jamming launches DOS attack on important channels such as control channel on which critical control messages are exchanged. To launch a channel-selective jamming attack, the adversary must be aware of the location of the targeted channel, which is defined by a separate frequency band, time slot, or PN code. To further improve the energy efficiency of selective jamming, the jammer can target specific packets of high importance. The important packets can be identified by overhearing the headers of the packets before the transmission is complete, or by anticipate based on protocol semantics. The adversary has to have inside information of the network in order to launch selective jamming attack.

The impact of jamming can propagate way beyond the physical jamming range

of an adversary, defined as the area within which packets are corrupted due to jamming. A sophisticated adversary can combine jamming with his knowledge of protocol specifics to impact different network layers. As an example, consider the implementation of a reactive routing protocol [40, 62] in multi-channel networks. Assume that nodes use a predefined channel to broadcast route request (RREQ) messages for the purpose of route discovery. Consider a jammer that attacks only RREQ messages, in order to partition the network or to route traffic through specific paths. In Figure 1.1(a), we show how the adversary can jam RREQ messages so that node n_i is unable to discover routes to any destination. In Figure 1.1(b), the adversary deploys jamming devices along a cut in the network which partition the network into two components A and B. In Figure 1.1(c), the jamming attack is intended to divert traffic to a particular link, thus forcing all traffic from A to B to flow through that link. This attack is similar to the sinkhole attack [42], in which a node attracts surrounding traffic by advertising the shortest route to a particular set of destinations. Once traffic is diverted, the adversary can control the flow of traffic from A to B by compromising a single node (either n_i or n_j). *Note that a jamming attack that targets the routing function cannot be prevented by existing secure routing protocols, as such protocols consider jamming outside the scope of their adversarial model* [37]. Similar intelligent attacks can be launched on critical network functionalities in other protocol layers. For example, the adversary may choose to jam the request-to-send (RTS) and clear-to-send (CTS) messages at the MAC layer so that the medium access delay is significantly increased.

1.1.2 Other attacks in wireless networks

In this section we introduce some well-known attacks in wireless network rather than jamming attack. The detailed discussion and countermeasures of these attacks are beyond the scope of this dissertation.

We will use the layered model to describe attacks at different layers of wireless network.

Physical layer attacks: Jamming is the primary attack at the physical layer.

Other than that, node tampering and destruction are the physical layer attacks in wireless sensor networks.

Link (MAC) layer attacks: Link layer threats include collision and interrogation [68]. A collision attack is similar to reactive jamming attack. In IEEE 802.11 protocol network, an adversary can use the NAV information to intentionally corrupt the ongoing frames. Interrogation attack can exhaust the target nodes's resource (e.g., battery) and consume network bandwidth by repeatedly sending handshake request (e.g., RTS) to elicit responses (e.g., CTS) from the target node.

Network layer attacks: There are many attacks targeting network layer of wireless network. Among them, spoofed, altered or replayed routing information (e.g., Hello flood attacks, routing cache poisoning attacks), sinkhole attacks, wormholes and homing attacks are well known attacks [41]. Hello flooding is an attack by broadcasting recorded hello packets from distant nodes in order to make nodes believe they are in the vicinity of distant nodes. When the victim node uses the distant nodes as next hop, they will find they can not reliably forward traffic. Routing cache poisoning attack takes advantage of the property of routing table updating, which is nodes updating their routing table based on overhearing packets. So when a malicious node wants to poison the routing table of neighboring nodes, it can broadcast a spoofed packet with certain nodes as destination and himself as the next hop, or even nonexistent nodes as next hop. Neighboring nodes overhearing this packet will add this route to the destination nodes in their route cache, which will make their routing unsuccessful or manipulated by the attacker.

Sinkhole (black hole) attack [4] makes a compromised node look especially attractive to surrounding nodes, for example by advertising itself having a better route to a destination, in order to lure the traffic from a particular area through this compromised node. Then the adversary can have many opportunities to manipulate the attracted traffic, such as dropping packets, selective forwarding, or even modifying packets. One way to combat sinkhole attack is by multipath routing, which sends the same packets over multiple paths in order to providing higher chance for destination to receive it. However, multipath routing increases power consumption

which is unpractical in energy sensitive applications such as sensor network.

In a wormhole attack, an adversary collects information at one point of the network (origin point), tunnels it to another point of the network (destination point) via a low-latency link to replay the information back into the network [64]. The nodes near the origin point will think they are one hop away from the nodes near the destination point, although they are out of communication range of each other. The error will propagate to more nodes near the two points. Then the nodes near the two points will become sinkhole without even being aware of that they are the victims of the wormhole attack [64]. The attacker can control and observe the lured traffic without the need to compromise network nodes.

Homing attack is the attack that identifies and targets the critical nodes (such as cluster head, node responsible for key management) by analyzing traffic pattern. The adversary then launch jamming attack or even physically destroy on these key nodes [68]. Possible solutions to combat homing attack is by using dummy packets to confuse the traffic pattern analysis [25].

Transport layer attacks: The attacks at transport layer exploit the end-to-end connection established by TCP like protocols. TCP SYN flood attack is a typical one. The adversary sends multiple connection requests to the victim node without completing the connections, in order to overwhelming the victim node's half-open connection buffer [68]. When connection is already established between legitimate nodes, the adversary can interrupt the active connection by sending packets with bogus sequence numbers [88].

Application layer attacks: The application layer communication is also vulnerable in terms of security. In sensor networks, the adversary can overwhelm certain node (such as the base station) by stimulating sensor nodes forward large volumes of traffic to it at the same time. This attack consumes network resource and node energy. Node impersonation or node replication attack is to add nodes with replicating IDs of existing nodes in the network. In this way the replication nodes manipulated by the adversary can misroute, corrupt or delete packets passing them.

1.2 Motivation of Dissertation

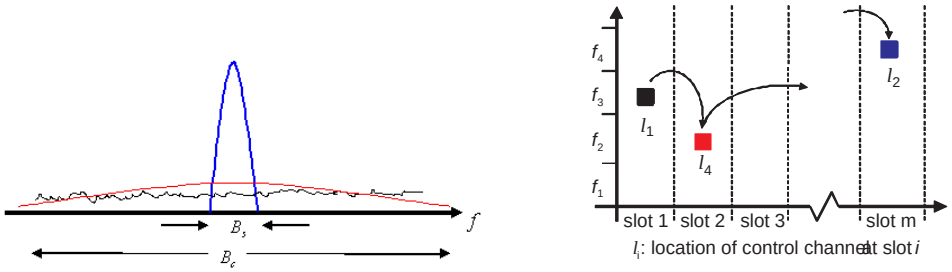
1.2.1 Control channel jamming attack

Organizing a collection of nodes into a wireless network requires cooperative implementation of critical network functions such as neighbor discovery, channel access and assignment, routing, and time synchronization. These functions are coordinated by exchanging messages on a broadcast channel, known as the *control channel*. In most network architectures, including mobile ad hoc, vehicular, sensor, cellular, mesh, and cognitive radio networks (CRNs), the location of the control channel, determined by its frequency band, time slot, or spreading code, is known a priori to all nodes participating in the network [3, 76].

From a security standpoint, operating over a globally known control channel constitutes a single point of failure. Networks deployed in hostile environments are susceptible to Denial-of-Service (DoS) attacks by adversaries targeting the functionality of the control channel [17, 47, 80]. If the adversary is successful, network service can be denied even if other available frequency bands remain operational. One of the most effective ways for denying access to the control channel is by jamming it. In fact, it was shown that jamming the control channel in GSM networks reduces the required power for performing a DoS attack by several orders of magnitude in [17, 80]. Control-channel jamming is particularly devastating for wireless ad hoc networks due to their cooperative nature. In such networks, the majority of network functions, including neighbor discovery and authentication, clustering, multiple access control, and routing, are actualized through the cooperation of all hosts in the network. Hence, control messages exchange among nodes within the same vicinity is frequent.

1.2.2 External adversarial model and internal threat model

Jamming in wireless networks has been primarily analyzed under an external adversarial model, in which the jammer has no knowledge of protocol specifics and cryptographic secrets [73, 74]. Conventional anti-jamming techniques rely exten-



(a) (b)

Figure 1.2: (a) Direct sequence spread spectrum (DSSSS), (b)frequency hopping spread spectrum (FHSS).

sively on spread spectrum (SS) communications, such as direct sequence spread spectrum (DSSS) and frequency hopping spread spectrum (FHSS), as shown in Figure 1.2. These techniques provide bit-level protection by spreading bits according to a secret PN code, known only to the communicating parties. An adversary unaware of this code has to transmit with a power which is several orders of magnitude higher compared to the SS transmission, in order to corrupt a SS signal. SS can protect communication from jamming attack as far as the PN code is kept secret from the adversary. However, if any of the network node gets compromised by the adversary and the PN code is revealed, a jamming adversary can deny communications by using very little energy. This can be particularly devastating for the control channel, which is by design a broadcast channel. In the case of broadcast communications, the sender's PN code must be shared by all (potentially non-trustworthy) receivers. The disclosure of such a secret due to the compromise of any receiver nullifies the gains due to SS. This situation is treated as the internal threat model, which is more difficult to address compared with jamming under external threat model.

In this dissertation we consider jamming attack especially control channel jamming attack under internal threat model, as well as the control channel establishment which is robust to jamming attack. The reasons are as following:

1: Control communication is critical to the subsequent data communication, so control channel can easily become the target of jamming adversary.

2: Network devices are relatively vulnerable to node capture and compromise in

wireless mesh network, sensor network and ad hoc network. The adversary can thus recovery the information stored in the hardware, such as the location of the control channel, PN code for spread spectrum, etc.

3: Internal attack can not be mitigated using only cryptographic method because the adversary already have access to the network secret, which makes this problem more sophisticated and interesting.

1.3 Main contribution and dissertation organization

The main contributions of this dissertation are listed as follows.

In Chapter 2, We consider a sophisticated adversary who exploits knowledge of protocol specifications along with cryptographic secrets to efficiently jam the control channel. This channel can be used by any layer in the protocol stack to broadcast control traffic, which could include coordination information needed for protocol operation in upper layers. To quantify the adversary's ability to deny access to the control channel, new security metrics are defined. A randomized distributed channel establishment and maintenance scheme is developed to allow nodes to establish a new control channel using frequency hopping. Under our scheme, network nodes are able to temporarily access a control channel until the jammer is removed from the network. Our method differs from classic frequency hopping in that no two nodes share the same hopping sequence. This allows for unique identification of compromised nodes by nearby ones. Our scheme is suitable for networks with static or dynamic spectrum assignment (e.g., CRNs). For the latter, we propose a modification of the original scheme to take into account the dynamic nature of channel availability in time and space. Assuming perfect random number generators, we analytically evaluate the proposed anti-jamming metrics. We verify our analytic results via extensive simulations. Both static spectrum and dynamic spectrum networks are considered and simulated.

In Chapter 3, We study the problem of anti-jamming broadcast communications

in the presence of inside jammers. We propose the *Time-Delayed Broadcast Scheme* (TDBS) for anti-jamming broadcast communications, based on FHSS communications. TDBS differs from classical FHSS designs in that two communicating nodes do not follow the same FH sequence, but are assigned unique ones that have high correlation properties. Unlike the typical broadcast operation where every receiver is eventually tuned to a common broadcast channel, TDBS implements the broadcast operation as a series of unicast transmissions spread both in frequency and time. To ensure resilience to inside jammers, the locations of the unicast transmissions, defined by a frequency band/slot pair, are only partially known to any subset of receivers. Because the jammer can only interfere with a limited set of frequency bands per time slot, a subset of the unicast transmissions are interference-free, thus propagating broadcast messages. The problem of FH sequence design, is mapped to a 1-factorization problem in complete graphs. TDBS is not meant to be a permanent replacement of the conventional broadcast mechanism in a benign setting. Broadcasting on a common frequency band achieves the optimal communication efficiency (one slot) in the absence of any jammer. TDBS is designed as an emergency mechanism for temporarily restoring communications until the jammer is physically removed from the network. Therefore, its primary focus is resilience to inside jammers and not the communication efficiency of the broadcast operation.

In Chapter 4, We develop cluster-based methods for control-channel assignment (CCA) in cognitive radio networks (CRN). This is an intuitive approach given the inherent partitioning of the network into clusters due to the location- and time-dependent spectrum availability. We formulate the clustering problem as a *bipartite graph problem*. In particular, we map the clustering process to the maximum edge biclique problem [24, 61] and the maximum one-sided edge cardinality problem [24]. Our mapping allows us to control the tradeoff between the set of common idle channels within each cluster and the cluster size. The cardinality of the set of common channels indicates the robustness to primary radio (PR) activity and jamming attack, since control-channel migration could be triggered when PR activity or jamming attack is detected on current control channel.

In Chapter 5, related work about jamming attacks in wireless network under external adversary model and internal threat model are discussed respectively. Control channel assignment problem in static spectrum network and dynamic spectrum network (CRN) are described in detail.

Finally, Chapter 5 draws some conclusions of this dissertation and suggests several topics for future research.

CHAPTER 2

Mitigating Control-Channel Jamming attacks with randomized distributed channel establishment scheme

2.1 Introduction

2.1.1 Motivation

Typically, jamming attacks have been analyzed and addressed as a physical-layer vulnerability. Conventional anti-jamming techniques rely extensively on spread spectrum (SS) [73]. These techniques provide bit-level protection by spreading bits according to a secret PN code, known only to the communicating parties. An adversary unaware of this code has to transmit with a power which is several orders of magnitude higher compared to the SS transmission, in order to corrupt a SS signal. However, in packet-radio networks, corrupting a few more bits than the correction capability of the error correcting code (ECC) (about 13% of the packet length for WLANs [60]) is sufficient to force the dropping of a data packet. Hence, the adversary need only stay active for a fraction of the time required for a packet transmission. Moreover, targeting the control channel, which typically operates at a low transmission rate, significantly reduces the adversary's effort. In fact, it was shown that the power required to perform a DoS attack in GSM networks is reduced by several orders of magnitude when the attack targets the control channel [17, 80]. Moreover, potential disclosure of cryptographic secrets (e.g., PN codes) by compromised nodes further reduces the adversary's effort. Note that because control information is broadcasted, PN codes must be shared by all intended receivers. The compromise of a single receiver leaves the network vulnerable to low-effort jamming attacks [17, 60, 80, 81]. In this article, we address the problem of *resisting control channel jamming in the presence of compromised nodes.*

In this section, we motivate our approach for establishing and maintaining the control channel in a cluster based architecture. Our method is based on the observation that the scope of control messages is typically confined to the range of the broadcaster (e.g., RTS/CTS messages). For multi-hop networks, broadcasted control messages can be relayed on the same or on a separate frequency band. Allocating different control channels to different neighborhoods within the same collision domain can potentially increase the control-channel throughput due to the reduction in interference between such neighborhoods. Moreover, allocating one unique channel for control has the following significant disadvantages: (a) a long-range transmission can jam the control channel for multiple neighborhoods, (b) the control channel re-establishment process has to be coordinated network-wide, and (c) the compromise of a single node reveals any shared PN codes used for broadcasting.

The impact of long-range jamming attacks can be significantly reduced by varying the control channel in space and time. Such a design also reduces the delay and communication overhead of the control channel re-establishment process, because it requires only local coordination. To mitigate the impact of jamming, we adopt a cluster-based architecture, where the network is partitioned into a set of clusters. Each cluster establishes and dynamically maintains its own control channel. In this design, it is sufficient to ensure that nodes can receive broadcast control messages from members of their own cluster, and that nodes at the boundaries of multiple clusters are aware of the control channels associated with these clusters. The control-channel establishment and maintenance process is facilitated by a clusterhead (CH) node within each cluster. CHs are regular nodes that are temporarily assigned with the responsibility of mitigating jamming, and can be periodically rotated. Several methods are readily available for organizing a wireless network into clusters and electing CHs [93].

In Fig. 2.1(a), we show an implementation of the control channel using one frequency. All nodes within the jammer's range are denied access to the control channel. In Fig. 2.1(b), we show a clustered approach where each CH is responsible for the establishment and maintenance of a separate control channel within its

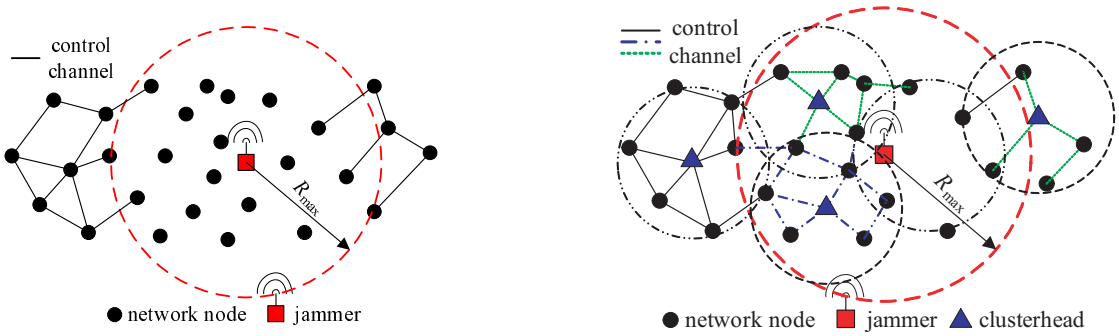


Figure 2.1: (a) The adversary blocks all control messages within range R_{max} by jamming a single frequency band, (b) the control channel is located at different channels within each cluster. The impact of the jammer is now confined to clusters within R_{max} that use the jammed channel.

cluster. The impact of the jammer is now confined to clusters within R_{max} that use the jammed frequency.

2.1.2 Main Contributions and Chapter Organization

We consider a sophisticated adversary who exploits knowledge of protocol specifications along with cryptographic secrets to efficiently jam the control channel. This channel can be used by any layer in the protocol stack to broadcast control traffic, which could include coordination information needed for protocol operation in upper layers. To quantify the adversary's ability to deny access to the control channel, new security metrics are defined. A randomized distributed channel establishment and maintenance scheme is developed to allow nodes to establish a new control channel using frequency hopping. Under our scheme, network nodes are able to temporarily access a control channel until the jammer is removed from the network. Our method differs from classic frequency hopping in that no two nodes share the same hopping sequence. This allows for unique identification of compromised nodes by nearby ones. Our scheme is suitable for networks with static or dynamic spectrum assignment (e.g., CRNs). For the latter, we propose a modification of the original scheme to take into account the dynamic nature of channel availability in time and space. Assuming perfect random number generators, we analytically evaluate the

proposed anti-jamming metrics. We verify our analytic results via extensive simulations. Both static spectrum and dynamic spectrum networks are considered and simulated.

The remainder of this article is organized as follows. In Section 2.2, we state the network and adversarial models, and propose new security metrics for evaluating control channel jamming. In Section 2.3, we present our randomized distributed scheme for maintaining control communications when the network is under attack. Section 2.4 describes the process of identifying compromised nodes. Analytical performance evaluation of our scheme is presented in Section 2.5. Section 2.6 concludes the chapter.

2.2 Model Description and Problem Formulation

2.2.1 Network Model

We consider a wireless ad hoc network. In the case of a static spectrum assignment, the network is assumed to operate over K orthogonal frequency bands. We use the terms frequency, frequency channel, or simply channel interchangeably to denote a separate frequency band. In the case of dynamic spectrum networks, the number of idle channels at time t, denoted by $K(t)$, varies according to primary radio (PR) activity. The maximum number of idle channels is equal to K. Cognitive radio (CR) nodes are capable of sensing the wireless medium and determining the set of idle channels at any given time. Various sensing methods can be used for this purpose [3].

Each node is equipped with a half-duplex transceiver. This is typical for most wireless devices equipped with a single radio[1]. We further consider a time-slotted system. Network nodes are assumed to be capable of slowly hopping between avail-

[1]Our schemes can benefit from a full-duplex transceiver design by exploiting concurrent transmissions/receptions of control information in multiple frequency bands, at the expense of increased hardware complexity. We leave the investigation of the properties and performance of our methods under a full-duplex communication model as future work.

able frequencies bands. For simplicity, we assume that one frequency hop can occur per time slot. Several messages may be exchanged during each slot. We assume that prior trust has been established between network nodes. Neighboring nodes share pairwise symmetric keys that can be used for secure communication and joint secret generation.

2.2.2 Adversarial Model

The goal of the adversary is to drop packets that are transmitted over the control channel. To do so, the adversary deliberately interferes with transmissions on selected frequency bands within a communication range R_{max}. Messages received by any node that is within the jamming range and at the jammed frequency band are assumed to be irrecoverably corrupted. Network nodes are assumed to be capable of detecting jamming attacks if they are within distance R_{max} from the jammer and are tuned to the jammed frequency band. Several methods are available for jamming detection [91], and any of them can be used for our purposes. We further assume that the adversary can physically compromise network devices and recover the content of their memory, including cryptographic secrets such as PN codes. He is also capable of hopping at the same rate as normal network nodes, thus jamming one channel per time slot (slow hopping jammer). This model is suitable when considering that the jammer is aware of the PN codes used for broadcasting. Therefore, he does not need to hop at a faster rate to jam the control channel. Note that with dedicated hardware, the jammer may be able to hop at a much higher rate than that of regular nodes. However, the jammer's hopping rate is limited by the time that he has to remain on a particular band in order to corrupt a sufficient number of bits from the targeted packet(s). Taking into account the interleaving function at the physical layer, this time can represent a significant portion of the slot duration [60].

2.2.3 Anti-Jamming Metrics

Numerous metrics have been proposed in the literature for evaluating jamming resilience. Traditional anti-jamming metrics such as the jamming-to-noise ratio and the jamming gain are mostly relevant under an external threat model. These metrics capture the amount of power needed by the adversary in order to interfere with legitimate transmissions at the physical layer [2, 60]. In our context, an adversary who is aware of a compromised PN code can follow that code in order to jam the control channel without significantly increasing his transmission power relative to the transmitted signal.

MAC layer metrics, such as the packet send ratio (PSR) and packet delivery ratio (PDR), were introduced by Xu et al. [91]. These metrics are useful for detecting a jamming attack, but are not reflective of the ability of our scheme in resuming the control channel operation. Our scheme aims at identifying the set of compromised nodes. This identification is critical for the re-establishment of the control channel. For this purpose, we define the following security metrics.

Definition 1. Evasion Entropy E_i–*Let I_i be a random variable that denotes the frequency of the control channel during slot i. We define the evasion entropy as:*

$$E_i = \mathrm{H}(I_i | I_{i-1}, I_{i-2}, \ldots I_0)$$

where $\mathrm{H}(X|Y)$ *is the conditional entropy of the random variable X given the random variable Y:*

$$\mathrm{H}(X|Y) \triangleq -\sum_{y}\sum_{x} \Pr[y] \Pr[x|y] \log_2 \Pr[x|y].$$

Here, $\Pr[y] = \Pr[Y = y]$ *and* $\Pr[x|y] = \Pr[X = x|Y = y]$.

The evasion entropy measures the uncertainty in the control channel location, given all previously observed locations and any internal knowledge due to node compromise.

Definition 2. Evasion Delay D–*The evasion delay is defined as the time between the successful jamming of the control channel and the re-establishment of a new one.*

Definition 3. Evasion Ratio *ER*–*The evasion ratio is defined as the fraction of time that the control channel is available for communication, in the presence of the jammer.*

2.3 Control Channel Implementation

Consider a given cluster, where each node is within the range of the CH. Suppose the current control channel is jammed by an adversary. The main idea behind our scheme is to have each node in the cluster hop between channels in a pseudo-random fashion, following a unique hopping sequence not known to other nodes. If the jammer captures the hopping sequence of a compromised node, then by design this node can be uniquely identified. After identification, the CH updates the hopping sequences of all nodes in the cluster except the compromised one. After this update, the effectiveness of a jammer who exploits knowledge from a compromised node becomes equivalent to the effectiveness of a jammer who hops randomly between channels. Note that our method is not a permanent solution for the control channel allocation, nor can it be used permanently for data communications due to its high communication overhead and delay. Rather, our scheme temporarily maintains control communication until the jammer and any compromised nodes are identified.

The hopping sequences assigned to various nodes are designed to overlap at certain time slots, which represent the control channel. These slots are kept secret. Given the uncertainty in the control channel location, control transmissions must be repeated in several slots to (probabilistically) ensure reception by the intended parties. Our scheme consists of five phases: (a) hopping sequence generation, (b) hopping sequence assignment, (c) control channel access, (d) compromised node identification, and (e) hopping sequence update. For dynamic spectrum networks, an intermediate step is applied to adjust the hopping sequences according to the current channel availability. We now describe each of the above phases.

2.3.1 Hopping Sequence Generation

By design, the hopping sequences assigned to different cluster members overlap only in a pre-defined number of slots, which are used to implement a broadcast control channel. In order to protect the secrecy of the control channel, the hopping sequences must satisfy the following properties: (a) *high evasion entropy*; knowledge of previous hops does not reveal any information about future ones, and (b) *high minimum Hamming distance*; when interpreted as codewords, any two sequences should have a high Hamming distance so that a compromised node can be identified.

Suppose that the cluster consists of n nodes plus the CH, and let the set of available channels be $\{1, \ldots, K\}$. To construct n hopping sequences of length $L+M$, where M denotes the number of slots implementing the control channel, the CH executes the following steps:

Step 1: Generate n random sequences s_j, $1 \leq j \leq n$, each of length L. For each sequence $s_j = \{s_j(1), \ldots, s_j(L)\}$, we have $\Pr[s_j(i) = k] = \frac{1}{K}$, where $k = 1, 2, \ldots, K$.

Step 2: Generate a *random channel location vector* $c = \{c(1), \ldots, c(M)\}$, where $\Pr[c(i) = k] = \frac{1}{K}$ for $i = 1, \ldots, M$, and $k = 1, 2, \ldots, K$.

Step 3: Generate a *random slot position vector* $v = \{v(1), \ldots, v(M)\}$, where $v(i) \in \{1, \ldots, L + M\}$, with $v(i) \neq v(j),\ \forall i \neq j$.

Step 4: In every sequence s_j, insert element $c(i)$ before element $s_j(v(i))$ to generate a new sequence m_j.

In Fig. 2.2, we show an example of the hopping sequence generation phase for three nodes, with $L = 12, M = 5$, and $K = 8$. Here, the indexes $\{1, \ldots, 8\}$ correspond to eight frequency bands $\{f_1, \ldots, f_8\}$. In Step 1, three random sequences s_1, s_2, and s_3 of length $L = 12$ are generated, with $s_j(i) \in \{1, \ldots, 8\}$. In Step 2, a random channel location vector c of length $M = 5$ is generated with $c(i) \in \{1, \ldots, 8\}$. This vector indicates the frequency bands of the control channel. In Step 3, the

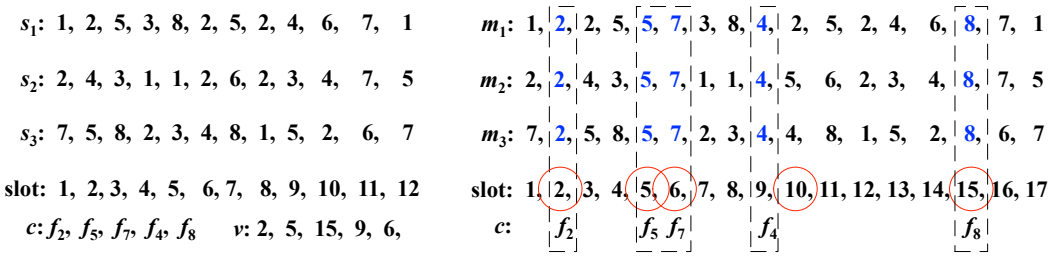


Figure 2.2: Hopping sequence generation for $L = 12, M = 5$ and $K = 8$. The control-channel location vector c is interleaved with the random sequences s_1, s_2, and s_3 at the slot positions indicated by the M-long vector v.

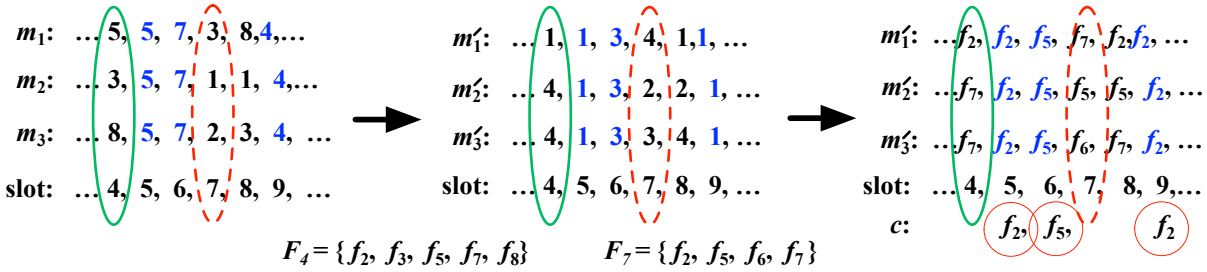


Figure 2.3: Adjusting the hopping sequences to account for dynamic channel availability.

random slot position vector v is generated. This vector indicates the five slots where the control channel is implemented. In Step 4, the sequences m_1, m_2, and m_3 are obtained based on s_1, s_2, s_3, c and v. Note that because the m_j's are a result of random interleaving of random sequences, *they are also random.* However, the sequences m_j, $1 \leq j \leq n$, are not mutually independent, because c is interleaved in specific slots of all sequences. Despite this fact, it still holds that knowledge of one sequence does not reveal any information regarding the other. This is due to the fact that the vector v, which indicates the slot locations where two hopping sequences overlap by design, is not known to non-CH nodes. Therefore, by knowing one hopping sequence m_j, one cannot predict the other sequence.

2.3.2 Generation for Dynamic Spectrum Networks

In dynamic spectrum networks, the set of channels available for use varies temporally and spatially. Consider a CRN. Suppose that the nodes are assigned hopping sequences m_j's, generated as in Section 2.3.1. Denote channel availability during time slot i by $\mathcal{F}_i = \{ch_i(1), ch_i(2), \ldots, ch_i(K(i))\}$, where $K(i)$ is the number of idle channels during slot i, $K(i) \leq K$. Here, $ch_i(j)$ denotes the index of the jth idle channel, $ch_i(j) \in \{f_1, \ldots, f_K\}$. The set of idle channels in each time slot can be determined by the underlying channel sensing process [3], with all nodes in a particular cluster agreeing on the same set [49]. However, two different clusters may have two different sets of idle channels. To adjust m_j to a hopping sequence m'_j for dynamic spectrum networks, each cluster node executes the following steps.

Step 1: Determine the channel availability set for time slot i : $\mathcal{F}_i$ $=\{ch_i(1), ch_i(2), \ldots, ch_i(K(i))\}$.

Step 2: Map index $m_j(i)$ to index $m'_j(i) = m_j(i) \pmod{K(i)} + 1$.

Step 3: Access frequency band $\mathcal{F}_i(m'_j(i))$.

The following example illustrates the above procedure. Consider three CRs that have been assigned the hopping sequences in Fig. 2.2. Suppose that for slot 4, the set of idle channels is $\mathcal{F}_4 = \{f_2, f_3, f_5, f_7, f_8\}$ $(K(4) = 5)$. CR_1 executes Step 2 above and computes $m'_1(4) = [m_1(4) \pmod{K_4})] + 1 = [5 \pmod 5] + 1 = 1$. In Step 3, CR_1 determines the next hop to be $\mathcal{F}_4(1) = f_2$. Similarly, CR_2 determines $m'_2(4) = [m_2(4) \pmod{K_4}] + 1 = [3 \pmod 5] + 1 = 4$, which denotes the 4th channel in the idle channel list, i.e., f_7. CR_3 hops to the same channel though its original index $m_3(4) \neq m_2(4)$. Suppose now that the set of idle channels for slot 7 has changed to $\mathcal{F}_7 = \{f_2, f_5, f_6, f_7\}$ (in reality, PR activity varies at a much slower rate compared to the scale of time slots). The CRs adjust their sequences to the current set of idle channels. The resulting sequences are shown in Fig. 2.3.

2.3.3 Hopping Sequence Assignment

The hopping sequences generated by the CH are assigned to individual cluster nodes via secure pairwise communication. Using pre-shared pairwise keys, the CH can establish pairwise shared PN codes with the members of its cluster. Note that the compromise of a cluster node only reveals the PN code shared between that node and the CH, while the rest of the pairwise PN codes remain secret. Thus, these codes can be used for jamming-resistant pairwise communication (but not for broadcasting of control information). The steps of the hopping sequence assignment for a node n_j are as follows:

Step 1: The CH and node n_j establish a pairwise PN code (this code can be either preloaded or generated based on a pairwise key K_{CH,n_j}).

Step 2: The CH provides n_j with the hopping sequence m_j, encrypted using the pairwise key K_{CH,n_j}. Message integrity is achieved through a message authentication code (MAC).

Step 3: Node n_j erases from its memory any information regarding the identity of the CH.

Step 3 ensures that after PN code assignment, the identity of the CH becomes a secret. Hence, an adversary who may later on compromise n_j cannot selectively target the compromise of the CH. Note that once hopping sequences are assigned, cluster nodes need not know the CH identity in order to access the control channel. In any case, the CH can prove his role to various nodes by using his knowledge of the PN codes that were assigned to individual nodes during the assignment phase. Any cluster node attempting to impersonate the CH would fail to "authenticate" itself, because it is not aware of the PN codes originally assigned.

2.3.4 Control Channel Access

The hopping sequences assigned to cluster members are designed to implement the control channel only during the slots indicated by the random slot position vector

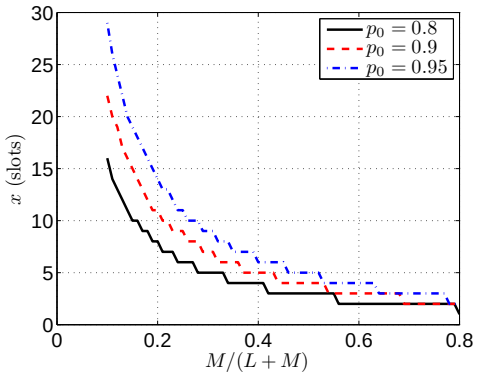


Figure 2.4: Number of slots required for accessing at least one control channel slot with probability p_0 as a function of the ratio $\frac{M}{L+M}$.

v. To prevent an adversary who compromises one cluster node from jamming the control slots, we require by design that v is not known to cluster nodes. Hence, nodes are not aware of which time slots solely belong to the control channel. To broadcast a control message, a node must repeat its transmission over consecutive slots. The goal here is to ensure that a control-channel slot is accessed at least once during this repetitive transmission. Let x denote the number of retransmissions of a control message. An appropriate value for x can be probabilistically computed based on the design parameters of the hopping sequences. That is, we can tune x such that a control-channel slot occurs with a desired probability p_0. The probability that the number of occurrences z of a control slot is larger than one in a total of x slots is

$$\Pr[z \geq 1] = 1 - \Pr[z = 0] = 1 - \left(\frac{L}{L+M}\right)^x. \tag{2.1}$$

Setting $\Pr[z \geq 1] \geq p_0$ and solving for x yields

$$x \geq \lceil \frac{\log(1-p_0)}{\log L - \log(L+M)} \rceil. \tag{2.2}$$

Fig. 2.4 depicts x versus $\frac{M}{L+M}$ for various values of p_0. We observe that for $\frac{M}{L+M} \geq 0.5$, fewer than 5 repetitions are required for all three values of p_0. For smaller values of $\frac{M}{L+M}$, the required number of slots x increases up to 28 slots when

$\frac{M}{L+M} = 0.1$ and $p_0 = 0.95$. The ratio $\frac{M}{L+M}$ controls the tradeoff between the efficiency of the broadcast communication and resilience to jamming under node compromise. A higher ratio decreases the necessary number of retransmissions for a successful broadcast, but also increases the time needed for the identification of a compromised PN sequence.

Note that several cluster nodes may want to broadcast a control message during the same control slot. Although we do not specify the MAC mechanism for coordinating access to this common slot, well-known multiple access techniques, ranging form pure random access to p-persistent CSMA protocols to CSMA with virtual carrier sensing, can be employed. Broadcast control messages are not acknowledged so as to avoid an ACK implosion situation [84]. This is in line with typical wireless protocols such as the 802.11 family. Therefore, a transmitting node does not know if its transmission was performed over a control-channel slot or whether the transmission attempt was successful. For this reason, the node must repeat such a transmission x times.

Upon the successful transmission of a broadcast control packet on a slot that belongs to the control channel, all nodes are able to correctly receive that packet. Since packets are transmitted/received in the context of a particular protocol/application/network function, they are accordingly passed on to upper layers of the network stack. Note that cluster nodes may receive multiple copies of the same control packet, because transmissions are repeated on multiple slots, and multiple sequences may coincide on slots other than the control channel slots. This replication of information is indicated by the inclusion of the same sequence number on the copies of the same packet (e.g., at the MAC layer header). That is, a node repeating the broadcast of the same control packet on multiple slots, keeps all fields of the packet identical. Hence, using the sequence number field, cluster nodes can reject multiple copies of the same control packet.

2.3.5 Hopping Sequence Update

Hopping sequences need to be updated when the CH detects that a node has been compromised. In this case, the CH is responsible for assigning new sequences to all uncompromised nodes. To do so, the CH synchronizes with the PN code of each individual node, prove his role as a CH, and assigns a new PN code. In detail, the following steps are executed:

Step 1: The CH synchronizes with PN code m_j (m_j is only known to the CH and n_j).

Step 2: The CH communicates to n_j a portion of m_j, which is meant to prove the CH's knowledge of m_j. This communication is secured by the pairwise key K_{CH,m_j}.

Step 3: The CH assigns a new sequence m_j^* to n_j. This communication is secured by the pairwise key K_{CH,m_j}.

Step 4: Node n_j erases all information regarding the identity of the CH.

The hopping sequence update phase differs from the initial hopping sequence assignment phase in the pairwise PN code used for communication. After the initial assignment, cluster nodes hop according to their m_j's. Hence, the CH has to follow each m_j to individually communicate with each node. Note that the compromise of node n_j does not reveal sequence $m_\ell, \ell \neq j$. Hence, the jammer cannot prevent the update of non-compromised nodes. The case of a CH compromise, which reveals all hopping sequences to the adversary, is addressed through CH rotation, as detailed in Section 2.4.3. Once a CH rotation has occurred, the new CH updates the hopping sequences of all cluster nodes by following the initial hopping sequence assignment process.

In Step 2, the CH proves his role to every cluster member that is assigned a new sequence. This step is necessary because information regarding the CH identity is erased after the initial hoping sequence assignment. Note that the pairwise key

shared between the CH and a cluster node n_j is not sufficient to authenticate the role of CH. Other cluster nodes may share pairwise keys with n_j. To avoid CH impersonation, the CH exploits his knowledge of the unique hopping sequences assigned to each node. When updating node n_j, the CH securely communicates part of the current sequence m_j (future hops) to n_j. Upon reception of a correct partial sequence, n_j will accept the sequence update performed in Step 3. After the successful assignment of m_j^*, node n_j erases the identity of CH from its memory (n_j is an uncompromised node and hence, will conform to Step 4). Steps 1-4 have to be repeated by every legitimate node in the cluster, leading to the isolation of the compromised node(s).

2.4 Identification of Compromised Nodes

In this section, we develop algorithms for the identification of compromised nodes. We first address the case of one compromised node, and then extend the treatment to multiple ones.

2.4.1 Compromise of a Single Node

Suppose that one cluster member n_j has been compromised. The adversary acquires the unique hopping sequence m_j assigned to this node. Because the slots implementing the control channel are secret, the adversary must follow m_j to efficiently jam the control channel. However, following m_j reveals the identity of the compromised node n_j. This identification is based on the Hamming distance between the sequences assigned to nodes and the jamming hopping sequence. In the following two propositions, we analytically evaluate the expected Hamming distance.

Proposition 1. *For two random and independently generated sequences m_j and m_ℓ, defined over an alphabet $\mathcal{A} = \{1, \ldots, K\}$, the expected Hamming distance* $\mathrm{E}[d(m_j, m_\ell)]$ *as a function of the sequence length X is given by*

$$\mathrm{E}[d(m_j, m_\ell)] = \frac{K-1}{K} X. \tag{2.3}$$

Proof. The proof is a direct consequence of the randomness and independence assumptions. Based on the sequence generation process outlined in Section 2.3.1, $\Pr[m_j(i) = k] = \frac{1}{K},\ \forall i$. Since the two sequences m_j and m_ℓ are assumed to be independent and random, they differ at slot i with probability

$$\Pr[m_j(i) \neq m_\ell(i)] = \frac{K-1}{K}. \tag{2.4}$$

The expected Hamming distance between two sequences of length X is equal to the expected number of successes in X such Bernoulli trials, i.e., $E[d(m_j, m_\ell)]= \frac{K-1}{K}X$. □

If the adversary has not compromised any node and is hopping according to a random sequence m_{jam}, the average Hamming distance between m_{jam} and any of the assigned sequences $m_j,\ 1 \leq j \leq n$, must increase at a rate of $\frac{K-1}{K}$. On the other hand, if m_{jam} is a subset of the sequence m_j of a compromised node n_j, the Hamming distance between m_{jam} and m_j is expected to be significantly lower (note that although the adversary may be aware of m_j, he may choose to follow only a subset of it to avoid being identified). The CH can exploit this observation to identify the compromised node.

In dynamic spectrum networks, the hopping sequences are not necessarily random, because of their adjustment to account for spectrum availability. Randomness is preserved only when the number of idle channels $K(i)$ is a factor of the alphabet size that was used to generate the original sequences. In the general case, the expected Hamming distance is expressed by Proposition 3.

Proposition 2. *Consider two random and independently generated sequences m_j and m_ℓ that are defined over an alphabet $\mathcal{A} = \{1, \ldots, K\}$. Suppose that the sequences are adjusted to m'_j and m'_ℓ, respectively, according to the process outlined in Section 2.3.2. The expected Hamming distance $E[d(m'_j, m'_\ell)]$ as a function of the length X of the sequences is*

$$\mathrm{E}[d(m'_j, m'_\ell)] = \left(1 - (K(i) - y_K) \cdot \left(\frac{x_K}{K}\right)^2 - y_K \cdot \left(\frac{x_K + 1}{K}\right)^2\right) \cdot X \tag{2.5}$$

where $x_K \triangleq \lfloor \frac{K}{K(i)} \rfloor$ *and* $y_K \triangleq [K \pmod{K(i)}]$.

Proof. According to Step 2 in Section 2.3.2, the hopping sequences are modified by a modulo $K(i)$ operation. The number of indexes of the original sequence that map to the same index in the modified sequence depends on the quotient of the division of K by $K(i)$, given by $x_K = \lfloor \frac{K}{K(i)} \rfloor$, and the remainder, given by $y_K = [K \pmod{K(i)}]$. In particular, for a modified sequence m'_j, it follows from elementary modulo arithmetic that

$$\Pr[m'_j(i) = w] = \begin{cases} \frac{x_K+1}{K}, & \text{if } 1 \le w \le y_k,\ y_k > 0. \\ \frac{x_K}{K}, & \text{if } y_k + 1 \le w \le K(i). \end{cases} \tag{2.6}$$

Let $\mathcal{M}$ be the event that two modified sequences m'_j and m'_ℓ match at slot i. Based on (2.6), we have

$$\Pr[\mathcal{M}] = \sum_{w=1}^{K(i)} \Pr[m'_j(i) = w, m'_\ell(i) = w] \tag{2.7a}$$

$$= \sum_{w=1}^{K(i)} \Pr[m'_j(i) = w] \Pr[m'_\ell(i) = w] \tag{2.7b}$$

$$= \sum_{w=1}^{y_k} \left(\frac{x_K + 1}{K}\right)^2 + \sum_{y_K+1}^{K(i)} \left(\frac{x_K}{K}\right)^2 \tag{2.7c}$$

$$= y_K \cdot \left(\frac{x_K + 1}{K}\right)^2 + \big(K(t_1) - y_K\big) \cdot \left(\frac{x_K}{K}\right)^2. \tag{2.7d}$$

Equation (2.7b) is due to the independence in the generation of the original sequences m_j and m_ℓ. Equation (2.7c) is due to the probability distribution in (2.6)

and Equation (2.7d) follows from the simplification of the sum. Given $\Pr[\mathcal{M}]$, it is easy to see that the expected Hamming distance for two sequences of length X is given by (2.5). □

Identification process: For identification purposes, the CH exploits his knowledge of the subsequences s_j, which are unique to individual nodes. Let s_{jam} denote the subsequence followed by the jammer, excluding the slot positions in vector v. To identify a compromised node, the CH measures the Hamming distance between s_{jam} and every assigned sequence s_j. Note that the half-duplex transceiver assumption limits the monitoring capabilities of the CH to a single channel per slot. Because the hopping sequence s_{jam} is not known in advance, the CH periodically tunes to s_j's of different nodes to compute the Hamming distance. To do so, the CH needs only to know if channel $s_j(i)$ was jammed at slot i. We now describe the steps for the identification process for a single compromised node. The pseudo-code is shown in Algorithm 1.

Step 1: Initialize $d(s_j, s_{jam}) = 0,\ \forall j$.

Step 2: Synchronize with the hopping sequence m_j of a randomly selected node n_j.

Step 3: For each slot i, $i \notin v$, if $m_j(i)$ is not jammed, set $d(s_j, s_{jam}) = d(s_j, s_{jam}) + 1$.

Step 4: After some number of slots $X \geq \gamma_0$, if $d(s_j, s_{jam}) < \mathrm{E}[d(s_j, s_{jam})] - \delta_X$, then node n_j is considered to be compromised.

Step 5: Randomly pick another node and repeat Steps 2-4 for X slots.

In Algorithm 1, each node is monitored for at least γ_0 slots to obtain an accurate estimate of $d(s_j, s_{jam})$.[2]. The tolerance margin δ_X is computed based on the stan-

[2]Faster identification of the compromised nodes can be achieved if the CH evaluates the Hamming distance of all sequences s_k for which $s_k(i) = s_j(i)$, on slot i, when the CH follows the sequence s_j. This method is expected to speed up the identification process by a factor of $\frac{1}{K}$.

Algorithm 1 Identification of a Single Compromised Node

1: $Initialize : d(s_j, s_{jam}) = 0,\ \forall j;\ j = 1;\ i = 0;\ \ CN \leftarrow \emptyset$
2: **while** J ==FALSE **do**
3: **for** $x = 1,\ \ x \leq X,\ \ x++$ **do**
4: **if** $m_j(i)$ NOT JAMMED & $m_j(i) \notin v$ **then**
5: $d(s_j, s_{jam}) = d(s_j, s_{jam}) + 1$
6: **end if**
7: **if** $d(s_j, s_{jam}) < \mathrm{E}[d(s_j, s_{jam})] - \delta_x$ && $x > \gamma_0$ **then**
8: J =TRUE, $CN \rightarrow n$, **break**
9: **else**
10: $i++$
11: **end if**
12: **end for**
13: **if** J ==TRUE **then**
14: **break**
15: **else**
16: $j++$
17: **end if**
18: **end while**CN

dard deviation σ_X of the Hamming distance. For example, considering $\delta_X = 3\sigma_X$ yields a 99.7% chance for the Hamming distance of two random sequences to be within that margin. The value of δ_X provides a tradeoff between the speed of identification (smaller δ_X yields tighter bounds on $\mathrm{E}[d(s_j, s_{jam})]$) and the false-positive identification rate In the case of static spectrum assignment, the standard deviation of $d(s_j, s_{jam})$ is equal to $\sigma_X = \sqrt{\frac{K-1}{K^2}X}$. The value of σ_X for dynamic spectrum networks can be easily computed based on Equation (2.6).

We emphasize that the value X takes into account only slots that belong to subsequences s_j. In reality, the delay in the identification of compromised nodes is larger due to the interleaving of common control slots that belong to c. The

latter slots do not contribute to the identification process and hence, are excluded from the computation. As an example, consider the hopping sequences shown in Fig. 2.2. Let $s_{jam} = s_2$, and $\gamma_0 = 10$. For $X = 10$, $\mathrm{E}[d(s_j, s_\ell)] = 8.75$, $\delta_X = 3$, and $\sigma_X = 3.1$. Initially, the CH follows s_1 for $X = 10$ slots. After the first ten slots, $d(s_1, s_{jam}) = 8$. The CH switches to sequence s_2. Because s_2 is the jamming sequence, $d(s_2, s_{jam}) = 0$. Thus, node n_2 is declared compromised. For this set of parameters, the jammer can avoid detection, only if he partially follows s_2 for a fraction $\alpha(X) \leq 40\%$ of the monitoring interval X. As X increases, the fraction $\alpha(X)$ converges to its expected value in the case of random jamming. This can be easily shown from the detection condition of Step 4 of the identification algorithm. Assuming an adversary which is active only for a fraction $\alpha(X)$ of the X slots corresponding to a compromised s_j and setting $\delta_X = \tau\sigma_X$, where τ denotes some desirable constant, the detection condition yields

$$\begin{aligned} d(s_j, s_{jam}) &= \mathrm{E}[d(s_j, s_{jam})] - \delta_X \Rightarrow \\ (1 - \alpha(X))X &= \frac{K-1}{K}X - \tau\sqrt{\frac{K-1}{K^2}X} \Rightarrow \\ \alpha(X) &= \frac{1}{K} + \tau\sqrt{\frac{K-1}{K^2 X}}, \end{aligned} \tag{2.8}$$

where we have used the fact that $d(s_j, s_{jam}) = (1 - \alpha(X))X$, when the jammer is following only a fraction $\alpha(X)$ of a compromised sequence s_j. From (2.8), it is evident that $\alpha(X) \to \frac{1}{K}$ when $X \to \infty$. This is a fairly intuitive result that indicates that with the progression of the monitoring period X, a jammer that partially follows a compromised PN code, cannot deviate from the behavior of a random jammer without being detected.

An implicit assumption of our identification process is that the CH is able to detect when a jamming signal interferes with the reception of a control message within his cluster. It is possible for the jammer to tune his transmission power so as to interfere with the reception at a cluster node, but not at the CH. Such a low-power jammer has a limited impact on the network due to his small jamming range. We are primarily concerned with the scenario presented in Fig. 2.1, in which

a high power jammer attempts to deny the control channel within a large network area.

2.4.2 Compromise of Multiple Nodes

When several nodes are simultaneously compromised, the jammer can combine the acquired hopping sequences to reduce the number of jammed slots. Without loss of generality, assume that nodes $n_1, \ldots, n_q$, $q < n$, are compromised. Suppose that the jamming sequence s_{jam} consists of the (time, frequency) pairs that are common to compromised sequences $\{m_1, \ldots, m_q\}$, excluding the slots in v. In the case of static spectrum networks, the expected length of s_{jam} is given by Proposition 4.

Proposition 3. *The expected length* $\mathrm{E}[X]$ *of a sequence* s_{jam} *consisting of the channel locations common to* q *random hopping sequences* $\{s_1, \ldots, s_q\}$ *of length* X *is*

$$\mathrm{E}[X] = \left(\frac{1}{K}\right)^q X. \tag{2.9}$$

Proof. The probability that q random sequences overlap at slot i is $\left(\frac{1}{K}\right)^q$. For the random sequences $\{s_1, \ldots, s_q\}$, the expected number of overlapping channel locations is expressed by the expected number of successes in repeating X Bernoulli trials with parameter $\left(\frac{1}{K}\right)^q$. □

Note that the adversary cannot differentiate between the slot positions assigned to the control channel and the $\left(\frac{1}{K}\right)^q L$ slot positions that match due to the s_j's. Hence, the adversary must jam all slots common to the compromised sequences to efficiently deny access to the control channel. For the case of dynamic spectrum networks, the expected length of the sequence s_{jam} is given by Proposition 4.

Proposition 4. *The expected length* $\mathrm{E}[X]$ *of a sequence* s_{jam} *consisting of the channel locations common to* q *hopping sequences* $\{s'_1, \ldots, s'_q\}$ *of length* X *is*

$$\mathrm{E}[X] = \sum_z \left(y_K \left(\frac{x_K + 1}{K}\right)^q + (K(z) - y_K)\left(\frac{x_K}{K}\right)^q \right) X_z \tag{2.10}$$

where z denotes the number of channel availability changes over the course of $X = \sum_z X_z$ slots, and X_z denotes the number of slots of the sequences s_j for which the number of idle channels is equal to $K(z)$.

Proof. The proof follows similar steps to the proof of Proposition 4 and is omitted. □

To identify compromised nodes, the CH correlates the random sequences $s_j,\ 1 \leq j \leq n$, with the jammed channel locations. The CH follows a monitoring sequence s_{CH}, which is a concatenation of subsequences from the s_j's.

$$s_{CH} = s_1(1:Y)||\ldots||s_n((n-1)Y+1:nY)$$

where Y denotes the number of slots that belong to the subsequences s_j and are devoted to monitoring a node. Note here that our computations ignore slots that belong to vector v. In reality, to monitor a subsequence s_j for Y slots, the CH must monitor n_j for $(1+\frac{M}{L+M})Y$ slots, on average. The CH maintains a matrix

$$A = \{a_{ji}|a_{ji} \in \{0,1\}\}_{n\times nY} \tag{2.11}$$

where each row j corresponds to node n_j and each column i corresponds to the ith slot. Note that only non-control slots are taken into account in the construction of matrix A. For A, an element $a_{ji} = 1$ if while residing on channel f during the ith slot, the CH detects f as jammed and $s_j(i) = f$. Otherwise, $a_{ji} = 0$. Considering each row $A(j)$ as a codeword, the CH computes the codeword weight $W(A(j))\ \forall j$, and ranks the weights in a descending order. The weight $W(C)$ of a binary codeword C is defined as the number of ones in the codeword. Compromised nodes are expected to have a significantly larger weight than other nodes and their weights will be of the same order. Assuming q compromised nodes, the expected weight for a codeword $A(j)$, $j = 1,\ \cdots,\ q$ is

$$\mathrm{E}[W(A(j))] = \left(\frac{1}{K}\right)^q qY \tag{2.12}$$

Algorithm 2 Identification of Multiple Compromised Nodes

1: $Initialize: A = 0,\ W = 0,\ j = 1,\ j$ =FALSE, $CN \leftarrow \emptyset$
2: v, n;
3: $m_{CH} = s_1(1:Y)||\ldots||s_n((n-1)Y+1:nY)$
4: **while** J ==FALSE **do**
5: **if** $m_{CH}(i)$ JAMMED & $i \notin v$ **then**
6: $a_{ji} = 1,\ W(j)++,\ \forall i,\ \exists\ s_{CH}(i) = s_j(i),\ i++$
7: **end if**
8: **if** $i > \gamma_1$ // sufficient sampling **then**
9: sort(W) // sort weights in a descending order
10: find $j,\ \exists\ W(A(j)) - \mathrm{E}[W(A(j)) > \delta_q$, $CN \leftarrow j$
11: J =TRUE
12: **end if**
13: **end while**CN

where E$[W(A(j))]$ is computed over the qY slots devoted to the monitoring of the q compromised nodes. The CH identifies the set of nodes with high weights and compares those weights to their expected value, expressed in (2.12). If the weight of a codeword exceeds the expected value by some margin δ_q, i.e., $W(A(j)) \geq \left(\frac{1}{K}\right)^q qY + \delta_q$, the corresponding node n_j is identified as compromised. The parameter δ_q is a tolerance margin related to the standard deviation of $W(A(j))$. The pseudo-code for the identification of multiple compromised nodes is shown in Algorithm 2.

During the execution of Algorithm 2, the CH is not aware of the number of compromised nodes q. Without knowing q, the CH compares the computed weight of each node with multiple threshold values δ_q, for different q's. If any node violates any threshold value, it is declared compromised and its revocation is initiated via a hopping sequence update.

2.4.3 Compromise of the Clusterhead

By compromising the CH, the adversary can obtain all sequences $s_j,\ 1 \leq j \leq n$, the corresponding sequences m_j, as well as c and v. Using his knowledge of c and v, the adversary can deny control-channel access to all cluster nodes by jamming only the control channel locations. To address such a strong attack, the role of the CH has to be periodically rotated among cluster members. The steps of the hopping sequence assignment in the case of a CH rotation are as follows:

Step 1 : The new CH randomly hops between channels.

Step 2 : In each slot, the CH attempts to communicate with a cluster node n_j to establish a pairwise shared PN code. Random hopping continues until the establishment of the PN code is confirmed by both parties (via an ACK message).

Step 3 : The CH assigns a new hopping sequence m_j^* to n_j, using the pairwise shared PN code. The sequence m_j^* conforms to the design outlined in Section 2.3.1.

Step 4 : Node n_j erases all information regarding the identity of the new CH.

Step 5 : Steps 1-4 are repeated until all legitimate nodes are assigned new hopping sequences, except for the previous CH.

When a CH rotation occurs, the new CH n_i has to update all cluster nodes except the previous CH with new PN codes. Because the new CH is not aware of the current PN sequences followed by each cluster node n_j, it randomly hops to different channels in order to meet each node and assign it a new hopping sequence. If n_i meets a node n_j, it first establishes a pairwise PN code with n_j (via a commonly derived seed) and then updates that node with m_j^*. For simplicity, we have made the assumption that one slot is sufficient for communicating m_j^* to n_j. In reality, several packets may be needed to do that. The communication between the new CH and any cluster node is still susceptible to jamming activity. However, because

the PN code used by the two parties is not known to the jammer, the transmission will eventually be successful. The reception of m_j^* is acknowledged by n_j via an acknowledgement message. The new CH repeats this process until all cluster nodes are assigned new PN codes and have acknowledged their reception.

Note that initiation of a CH rotation has to be invoked by the individual cluster nodes, since the current CH is compromised. Nodes can declare the CH to be compromised if they cannot access the control channel for a prolonged period of time (computed in Section 2.5.4). Any cluster node other than the current CH may decide to become the CH and initiate the CH rotation process. If more than one nodes decide to become CHs, a cluster may be partitioned to smaller clusters.

2.5 Performance Evaluation

In this section, we analytically study the anti-jamming metrics introduced in Section 2.2, and validate our analysis via simulations. We evaluate both static and dynamic spectrum networks.

Simulation Setup: For static spectrum networks, we construct the hopping sequences m_j according to the process described in Section 2.3.1. Under an external jammer model, the adversary jams channels in a random fashion. Under an internal jammer model, the adversary jams only those slots in which the compromised sequences overlap, and remains silent in all other slots. The simulations are run for 5,000 slots.

For dynamic spectrum networks, we simulate PR activity to obtain temporally varying spectrum availability. We consider a cellular network as the primary network (PRN), operating over $K = 10$ frequency bands. The call arrival process at the PRN follows a Poisson distribution with an arrival rate of $\lambda = 2$ calls/min. The call duration is assumed to be exponentially distributed with parameter μ. For each value of μ, we run the simulation until 5,000 calls are completed. The set of idle channels is updated each time a new call arrives, or when a call is terminated. The jammer is assumed to be aware of the set of idle channels at every slot. The slot

duration is set to 100 msec. Each secondary node (e.g., CR node) dynamically adjusts its hopping sequence according to the process described in Section 2.3.2.

2.5.1 External Jammer

We first consider the case of an external jammer. In this scenario, the hopping sequences followed by each cluster node remain secret. Before we compute the metrics of interest, we derive the optimal jamming strategy for an external jammer. Without any inside information, the jammer must guess the location of the control channel. The optimal jamming strategy is obtained from the following proposition.

Proposition 5. *The optimal strategy of an external jammer is to continuously jam the most frequently visited channel.*

Proof. Let c_{jam} denote the subsequence of m_{jam} corresponding to the locations of control channel slots; i.e., $c_{jam} = \{m_{jam}(i) : i \in v\}$ (v denotes the random slot position vector). Let also $\mathcal{P} = \{p_1, p_2, \ldots, p_K\}$ and $\mathcal{Q} = \{q_1, q_2, \ldots, q_K\}$ denote the probability distribution functions from which values $c(i)$ and $c_{jam}(i)$ are drawn, respectively. $\mathcal{Q}$ is optimal when the expected Hamming distance $\mathrm{E}\left[d(c, c_{jam})\right]$ is minimized, i.e., the jammer is able to overlap with c in the maximum number of slots. Suppose that $\pi = \{\pi(1), \ldots, \pi(k)\}$ is a permutation of the set of channels $\{1, \ldots, K\}$ such that $p_{\pi(1)} \geq \ldots \geq p_{\pi(K)}$. That is, the discrete probabilities of $\Pr[c(i) = k]$ are arranged in descending order. The probability that c and c_{jam} overlap at index i (which corresponds to slot $v(i)$) is

$$\begin{aligned} \Pr[c(i) = c_{jam}(i)] &= \sum_{j=1}^{K} \Pr[c(i) = \pi(j), c_{jam}(i) = \pi(j)] \\ &= \sum_{j=1}^{K} p_{\pi(j)} q_{\pi(j)} \end{aligned} \tag{2.13}$$

For a sequence of length X, the expected Hamming distance between c and c_{jam} is $\mathrm{E}[d(c, c_{jam})] = (1 - \Pr[c(i) = c_{jam}(i)])X$ (overlapping in two different slots are

independent events). Hence, the expected Hamming distance is minimized when (2.13) is maximized.

Maximization of (2.13) can be shown as follows. Consider two distributions $\mathcal{P} = \{p_1, p_2, \ldots, p_K\}$ and $\mathcal{Q} = \{q_1, q_2, \ldots q_K\}$, and also consider two cases for the distribution $\mathcal{Q}$: $\{q_{\pi(1)}, q_{\pi(2)}, \ldots, q_{\pi(K)}\} = \{1, 0, \ldots, 0\}$ and $\{q'_{\pi(1)}, q'_{\pi(2)}, \ldots q'_{\pi(K)}\}$ with $q'_{\pi(1)} < 1$. Let $S = \sum_{j=1}^{K} p_{\pi(j)} q_{\pi(j)}$ and $S' = \sum_{j=1}^{K} p_{\pi(j)} q'_{\pi(j)}$. Then,

$$\begin{aligned} S' - S &= \sum_{j=1}^{K} p_{\pi(j)} q'_{\pi(j)} - \sum_{j=1}^{K} p_{\pi(j)} q_{\pi(j)} \\ &= \sum_{j=1}^{K} p_{\pi(j)} q'_{\pi(j)} - p_{\pi(1)} \cdot q_{\pi(1)} \\ &\leq \sum_{j=1}^{K} p_{\pi(1)} q'_{\pi(j)} - p_{\pi(1)} \\ &= p_{\pi(1)} \sum_{j=1}^{K} q'_j - p_{\pi(1)} \\ &= 0. \end{aligned}$$

Hence, $\sum_{j=1}^{K} p_{\pi(j)} q_{\pi(j)}$ is maximized when the distribution $\{q_{\pi(1)}, q_{\pi(2)}, \ldots q_{\pi(K)}\}$= $\{1, 0, \ldots, 0\}$. □

When the channel location vector c is random, the jammer is expected to have the same likelihood of success, regardless of how he constructs m_{jam}. This can be easily seen from (2.13), when $p_1 = \ldots = p_K = p$. Note that for the subsequence c_{jam} which is of interest, it holds that $\Pr[c(i) = c_{jam}(i)] = p$, irrespective of the values of q_i's. If c is not random (this is true for dynamic spectrum networks where the modulo operation reduces the randomness in c), the optimal jamming strategy is to continuously jam the most probable channel. Based on equation (2.6), in the case of dynamic networks channels $\{1, \ldots, y_K\}$ ($y_K = [K \pmod{K(i)}] > 0$) occur in the hopping sequences m'_j with the highest probability. Therefore, the optimal jamming strategy is to continuously jam any of the channels in $\{ch_i(1), \ldots, ch_i(y_K)\}$, yielding a success probability of $\frac{\lfloor \frac{K}{K(i)} \rfloor + 1}{K}$ per slot. In fact, choosing any probability

distribution which distributes the probability mass on the set $\{ch_i(1), \ldots, ch_i(y_K)\}$ yields the same probability of success. Given the optimal jamming strategy, we now evaluate the proposed anti-jamming metrics for an external jammer.

Evasion Entropy

The elements $m_j(i)$ of a sequence m_j are generated independently for each slot. Hence, knowledge of previous control channel locations does not reveal any information about future ones. In this case, $E_i = H(I_i)$. For static spectrum networks, $m_j(i)$ is drawn from a uniform distribution, yielding the maximum value for the evasion entropy, i.e., $E_i = \log_2 K$ bits. For dynamic spectrum networks, E_i depends on the number of idle channels $K(i)$. Using the probability distribution of eq. (2.6), it can be shown that

$$E_i = \frac{1}{K}\left(\log_2\left(\frac{K^{y_K - K(i)x_K}}{(x_K+1)^{(x_K+1)y_K} x_K^{(K(i)-y_K)x_K}}\right)\right) \tag{2.14}$$

where $x_K \triangleq \lfloor \frac{K}{K(i)} \rfloor$ and $y_K \triangleq [K \pmod{K(i)}] > 0$.

Evasion Delay

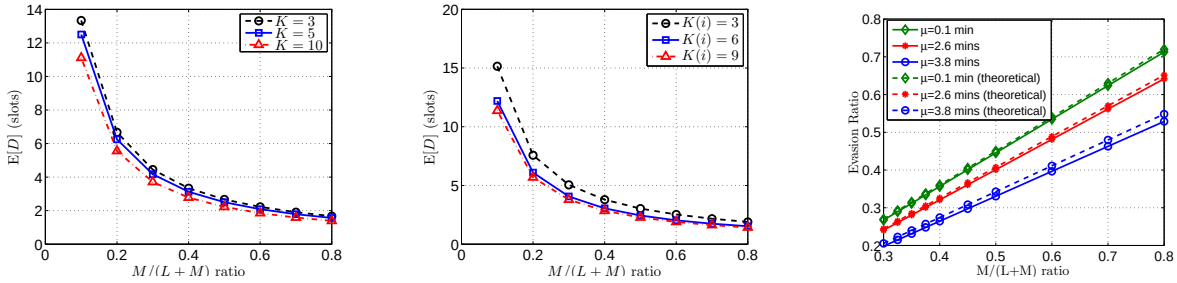
Proposition 6. *In static spectrum networks, the expected evasion delay* $\mathrm{E}[D]$ *for re-establishing the control channel when no node has been compromised is*

$$\mathrm{E}[D] = \frac{K}{K-1} \cdot \frac{L+M}{M}. \tag{2.15}$$

Proof. $\mathrm{E}[D]$ is equal to the expected number of required slots $\mathcal{N}$ before the control-channel slot occurs for the first time, times the number of tries $\mathcal{R}$ needed to evade jamming. Thus,

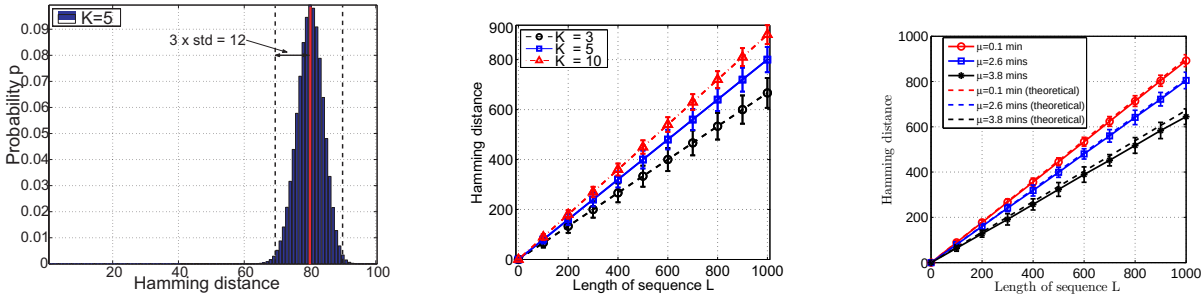
$$\mathrm{E}[D] = \mathrm{E}[\mathcal{R}\mathcal{N}] = \mathrm{E}[\mathcal{R}]\mathrm{E}[\mathcal{N}]. \tag{2.16}$$

Note that $\mathcal{R}$ and $\mathcal{N}$ are independent random variables. The probability of evading jamming for random hopping sequences, assuming an optimal jamming strategy, is equal to $\frac{K-1}{K}$. Thus, $\mathrm{E}[\mathcal{R}] = \frac{K}{K-1}$. By construction, slot i is a control-channel slot



(a) (b) (c)

Figure 2.5: (a) E[D] as a function of the ratio $\frac{M}{L+M}$ for static spectrum networks, (b) E[D] as a function of the ratio $\frac{M}{L+M}$ for dynamic spectrum networks, (c) E[ER] as a function of $\frac{M}{L+M}$ for dynamic spectrum networks.



(a) (b) (c)

Figure 2.6: (a) pmf of the Hamming distance between two random sequences of length 100, (b) expected Hamming distance as a function of a sequence of length L for static spectrum networks (error margins denote 99.7% confidence intervals), (c) expected Hamming distance as a function of L for dynamic spectrum networks.

with probability $\frac{M}{L+M}$. Therefore, the first re-occurrence of the control channel follows a geometric distribution with parameter $\frac{M}{L+M}$, and $\mathrm{E}[\mathcal{N}] = \frac{L+M}{M}$. Substituting $\mathrm{E}[\mathcal{R}]$ and $\mathrm{E}[\mathcal{N}]$ into (2.16) completes the proof. □

For the case of dynamic spectrum networks, the probability of evading jamming in slot i is equal to $(1 - \Pr[\mathcal{M}])$, where $\Pr[\mathcal{M}]$ is given in eq. (2.7d). Therefore, $\mathrm{E}[\mathcal{R}] = \frac{1}{1-\Pr[\mathcal{M}]}$, whereas $\mathrm{E}[\mathcal{N}]$ remains the same as in static networks. Substituting $\mathrm{E}[\mathcal{R}]$ and $\mathrm{E}[\mathcal{N}]$ yields

$$\mathrm{E}[D] = \frac{1}{1 - \Pr[\mathcal{M}]} \cdot \frac{L+M}{M}. \tag{2.17}$$

Evasion Ratio

The evasion ratio reflects the communication efficiency of the control channel. It measures the fraction of slots used for control communication in the presence of a jammer. The expected value of the evasion ratio E[ER] can be directly derived by taking the inverse of the evasion delay.

Simulation and Numerical Examples

In Fig. 2.5(a), we show the expected evasion delay as a function of the ratio $\frac{M}{M+L}$ for static spectrum networks. The ratio $\frac{M}{M+L}$ denotes the fraction of time devoted to the control channel. It can be observed that the evasion delay drops with the increase in $\frac{M}{M+L}$. This is due to the fact that the control channel occurs more frequently and hence, the jammer will be unsuccessful in guessing the location of the control channel in fewer slots. However, in the event of a node compromise, fewer slots are available to identify compromised sequences when $\frac{M}{M+L}$ increases. In Fig. 2.5(b), we show the expected evasion delay as a function of $\frac{M}{M+L}$ for various values of $K(i)$ and for dynamic spectrum networks. This graph corresponds to equation (2.17). For a fixed value of $K(i)$, a behavior similar to the case of static spectrum networks is observed.

The evasion ratio can be obtained by inverting the values of the evasion delay. E[ER] increases linearly with $\frac{M}{M+L}$. To take into account temporal variations in spectrum availability in the case of CRNs, we compute the evasion ratio under simulated PRN activity. In Fig. 2.5(c), we show the evasion ratio as a function of $\frac{M}{L+M}$ for dynamic spectrum networks. Solid lines correspond to the simulation values, while dashed lines correspond to the theoretical ones. To obtain the theoretical values, we used Equation (2.17) to calculate the evasion delay and then computed its inverse value. For the calculation of Pr[$\mathcal{M}$], the mean value of the number of idle channels $E[K]$, obtained via simulation, was used. For the simulation results, we assumed the adversary is aware of the set of idle channels $\mathcal{F}_i$ in each slot i. Based on the optimal jamming strategy, the adversary jams the most probable channel in

each slot. If the adversary succeeds in jamming the control channel in slot i, we measure the delay until the control channel is re-established. The evasion ratio is computed as the inverse value of the average evasion delay. From Fig. 2.5(c), we observe that the simulated values closely match the theoretical ones. As expected from the theoretical analysis, the evasion ratio is a linear function of the fraction of time that the control channel is available.

2.5.2 Compromise of a Single Node

When a single node n_j is compromised, its hopping sequence m_j is revealed to the adversary. By following m_j, the adversary can jam all slots implementing the control channel. In this case, $E_i = 0$ and $\mathrm{E}[ER] = 0$, for as long as the compromised node is undetected. The evasion delay is equal to the time required to identify the compromised node. Under a single compromised node scenario, we evaluate the properties of the Hamming distance between randomly hopping sequences and correlated ones, that lead to the identification of the compromised node.

In Fig. 2.6(a), we show the pmf of the Hamming distance between two random sequences when an alphabet $\mathcal{A} = \{1, \ldots, 5\}$ is used for random sequence generation. The pmf is concentrated in a small region around the mean. For a sequence of length 100, 99.7% of possible random sequences are expected to have a Hamming distance of at least 68. If a jammer overlaps with m_j in more than 32 slots per 100, the CH will declare m_j to be compromised.

In Fig. 2.6(b), we show the expected Hamming distance as a function of the sequence length L for static spectrum networks. Error margins indicate the 99.7% confidence interval (three standard deviations). We observe that the event of node compromise can be easily identified by comparing the hopping sequence of the jammer to those assigned to cluster nodes. The Hamming distance must fall within well-confined margins, allowing fast identification of the compromised node.

In Fig. 2.6(c), we show the expected Hamming distance as a function of L for dynamic spectrum networks. Both theoretical and simulation values are shown. Note that the temporal variations in channel availability do not significantly affect

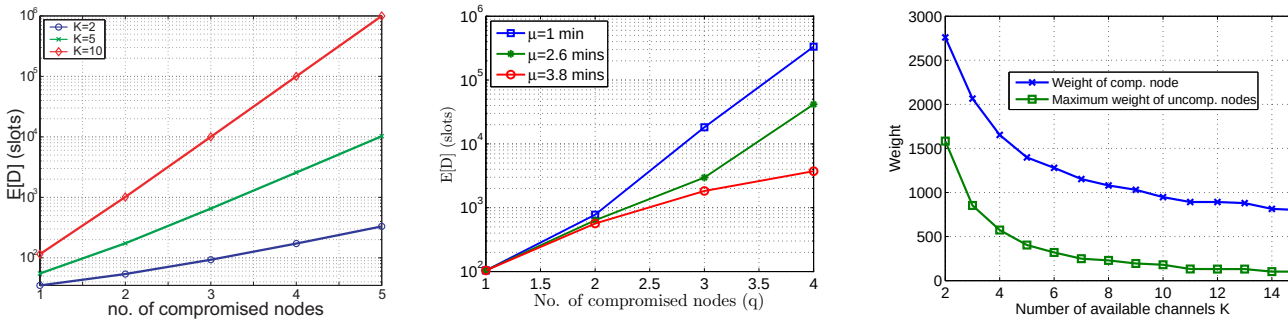


Figure 2.7: (a) E[D] as a function of the number of compromised nodes for static spectrum networks, (b) E[D] as a function of the number of compromised nodes for dynamic spectrum networks, (c) weight of the compromised node compared to the maximum weight of uncompromised ones.

the expected Hamming distance, which increases linearly with the length of the hopping sequences. The allowable deviation from the expected value remains small, leading to fast identification if a jammer follows a compromised sequence.

2.5.3 Compromise of Multiple Nodes

When multiple nodes are compromised, the jammer can combine their hopping sequences to obtain the channel locations of the slots in which these sequences overlap. This reduces the adversary's effort to jam the control channel (fewer slots need to be jammed) and makes the identification process more difficult. As in the case of a single compromised node, $E_i = 0$ and E[ER] $= 0$. To evaluate the evasion delay, we compute the time required to identify the set of compromised nodes, assign new sequences to uncompromised ones, and re-establish the control channel.

Suppose that q nodes are compromised in a cluster of n nodes. According to Algorithm 2, each of the q nodes is expected to have a weight of $\left(\frac{1}{K}\right)^q qY$, where Y is the time in slots that the CH uses to monitor each of the q compromised nodes. Let γ_0 be the number of *jammed* slots that are required for identification of the compromised nodes. To observe γ_0 jammed slots, the CH needs to monitor channels according to m_{CH} for an average time of $qX = K^q\gamma_0$ slots. Upon identification of the compromised nodes, the CH must assign new hopping sequences to the remaining

$(n-q)$ uncompromised nodes, yielding an additional delay of $(n-q)X_c$ slots, where X_c is the number of slots needed to assign a new sequence. Once sequences are assigned, a delay equal to the first occurrence of the control channel under a random jammer is incurred. Thus, the total expected evasion delay is:

$$E[D] = K^q\gamma_0 + (n-q)X_c + \frac{K}{K-1} \cdot \frac{L+M}{M}. \tag{2.18}$$

In Fig. 2.7(a), we show E[D] as a function of q for static spectrum networks. For simplicity, we take $X_c = 1$ and $\gamma_0 = 100$ slots. We observe that for large values of K, E[D] is very large when $q \geq 3$. This is due to the fact that the probability of overlapping among the q sequences at random becomes very small for large K. Thus, a much longer observation period is required to identify compromised nodes. For faster identification, the CH may limit the assigned hopping sequences to a subset of $\mathcal{M}$.

Fig. 2.7(b) shows E[D] as a function of q for dynamic spectrum networks. We observe a low value of E[D] when the PR activity is high. This behavior can be explained as follows. High values of μ translate into a smaller number of idle channels $K(i)$. Therefore, CRs hop between a smaller set of channels. The probability of compromised sequence overlap in a slot i, $i \notin v$, a necessary condition for their identification, increases with the reduction in $K(i)$. This is also evident from (2.18). Hence, the CH is able to identify compromised nodes faster using Algorithm 2.

To verify the effectiveness of Algorithm 2, we perform the following simulation experiment. We consider a cluster of 10 nodes and generate 10 hopping sequences, each of length 5,000. q of those sequences are assumed to be compromised. The jammer computes the jamming sequence m_{jam} as the intersection of the compromised sequences, and jams only the slots in which the q sequences overlap. Algorithm 2 is executed to compute the weight of each hopping sequence. The CH monitors the K channels according to sequence m_{CH}, following each sequence in a round-robin manner for 100 slots. In Fig. 2.7(c), we show the average weight E[W] of a compromised node compared with the maximum weight obtained from the set of uncompromised sequences, as a function of K. We observe that compromised nodes have a con-

sistently higher weight than uncompromised nodes, leading to identification of the former ones. Fig. 2.8 shows a comparison between the weight of compromised nodes and the maximum weight of uncompromised nodes, as a function of q.

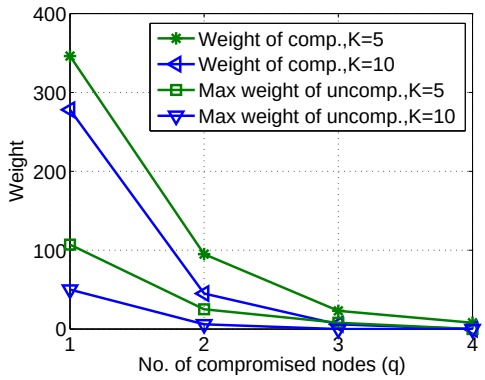


Figure 2.8: Average weight of compromised nodes and maximum weight of uncompromised ones versus q.

When the number of compromised nodes is small (less than 4), the weight of a compromised node is sufficiently distinct from the weight of an uncompromised node. However, for higher values of q, compromised sequences have less probability to coincide in each slot except the control channel slots. In this scenario, the CH must monitor each node for a large number of slots, in order to measure disparities in the weights of different sequences.

2.5.4 Compromise of the Clusterhead

If the CH is compromised, the adversary knows the hopping schedules of all nodes in the cluster as well as the slots of the control channel. Hence, the evasion entropy and the evasion ratio are equal to zero. The evasion delay E[D] is equal to the sum of three components: (a) the delay E[D_1], until the compromise of the CH is detected, (b) the delay E[D_2], of assigning new hopping sequences to cluster members, and (c) the delay E[D_3]. for re-establishing the control channel.

Cluster nodes consider the CH compromised when $E[ER]$ falls below a threshold value ρ_0 for an extended period of time. The parameter ρ_0 is fixed and depends on

the expected delay under a fixed number of compromised nodes. Let q_0 be the maximum tolerable number of compromised nodes within a cluster, before the CH is assumed to be compromised. The evasion delay when q_0 nodes are compromised ($E[D_1]$ in the calculation of $E[D]$) is given by equation (2.18), with $q = q_0$. The computation of $E[D_1]$ is based on fixed system parameters such as γ_0, K, L, M, and X_c. Taking the inverse of $E[D_1]$ when $q = q_0$, yields the threshold value ρ_0 that triggers a CH rotation. To detect the compromise of the CH, individual nodes compare $E[ER]$ with ρ_0.

Proposition 7. *The expected delay until the new CH assigns new hopping sequences to $n-1$ cluster nodes (excluding the compromised CH) is*

$$\mathrm{E}[D_2] = \frac{K^2}{K-1}(n-1)X_c. \tag{2.19}$$

Proof. Once the CH is considered compromised, all cluster nodes hop according to self-generated random sequences. Let m_{CH} denote the hopping sequence of the new CH. The CH succeeds in communicating with node n_j at slot i if $m_{CH}(i) = m_j(i)$ and $m_{CH}(i) \neq m_{jam}(i)$. Given that the sequences m_j and m_{CH} are random,

$$\Pr[m_j = m_{CH}, m_j \neq m_{jam}] = \frac{1}{K}\frac{K-1}{K} = \frac{K-1}{K^2}. \tag{2.20}$$

The number of slots until the first success is geometrically distributed with mean of $\frac{K^2}{K-1}$. The CH has to repeat the same process for all $n-1$ cluster nodes (the compromised CH is excluded from the hopping sequence update process). Assuming that X_c time slots are needed for the assignment of the new sequence, the expected delay $E[D_2]$ until all cluster nodes have received a new hopping sequence is equal to $\frac{K^2}{K-1}(n-1)X_c$. □

With the assignment of new sequences, the adversary's success becomes equivalent to that of an external jammer. An additional delay $\mathrm{E}[D_3]$ is incurred until a slot implementing the control channel occurs in the new hopping sequences. This delay is equal to the evasion delay in the case of the external jammer. The values

of $\mathrm{E}[D_2]$ and $\mathrm{E}[D_3]$ are negligible compared with $\mathrm{E}[D_1]$, given that $\mathrm{E}[D_1]$ grows exponentially with the number of compromised nodes, whereas $\mathrm{E}[D_2]$ and $\mathrm{E}[D_3]$ are constant. Hence, the expected value for the evasion delay under CH compromise approximates the expected evasion delay as calculated in (2.18) for the maximum acceptable value of q.

2.6 Conclusions

We addressed the problem of control-channel jamming attacks from insider nodes. We proposed a randomized distributed scheme for maintaining and establishing a broadcast channel using frequency hopping. Our method differs from classical frequency hopping in that the communicating nodes are not synchronized to the same hopping sequence. Instead, each node follows a unique hopping sequence. We further proposed a mechanism for adjusting hopping sequences to dynamic spectrum conditions without incurring any extra overhead. Our scheme can identify compromised nodes through their unique sequences and exclude them from the network. We evaluated the performance of our scheme both in static- and dynamic-spectrum networks, based on the metrics of evasion entropy, evasion delay, and evasion ratio. We further evaluated the Hamming distance between the jamming sequence and those assigned to compromised and uncompromised nodes. Our proposed scheme can be utilized as a temporary solution for re-establishing the control channel until the jammer and the compromised nodes are removed from the network.

CHAPTER 3

Thwarting Inside Jamming Attacks on Wireless Broadcast Communications

3.1 Introduction

3.1.1 Motivation

Wireless communications are vulnerable to intentional interference attacks, typically referred to as jamming. In the simplest form of jamming, the adversary interferes with the signal reception by transmitting a continuous jamming waveform [73] or several short jamming pulses [59]. Conventional anti-jamming techniques rely extensively on spread spectrum (SS) communications, such as direct sequence spread spectrum (DSSS) and frequency hopping spread spectrum (FHSS) [2, 73]. SS provides bit-level protection by spreading bits according to a secret pseudo-random noise (PN) code, known only to the communicating parties. In the case of broadcast communications, the sender's PN code must be shared by all (potentially non-trustworthy) receivers. The disclosure of such secrets due to the compromise of any receiver nullifies the gains due to SS [54, 65].

Several researchers have studied the problem of anti-jam-ming broadcast communications under an internal threat model [8, 22, 48, 54, 65, 66, 80, 81]. Methods in [8, 48, 54, 66] eliminate the dependency of broadcast on shared secrets. Baird et al. proposed the encoding of "indelible marks" at specific locations within each broadcasted message [8]. Assuming that an active jamming attack cannot flip a bit '1'to a bit '0', it was shown that a jammer cannot erase packets from the wireless channel (but can inject arbitrary packets). Pöpper et al. [65] proposed a method called Uncoordinated DSSS (UDSSS), in which broadcast transmissions are spread according to a PN code, randomly selected from a public codebook. At the receiving

end, nodes decode received messages by exhaustively applying every PN code in the public codebook. Liu et. al. proposed RD-DSSS, a randomized differential DSSS scheme that also relies on randomly selected PN codes [54]. Compared to UDSSS, the RD-DSSS scheme provides resilience to reactive jammers.

Note that when the spreading PN code is not known a priori, broadcast transmissions must be repeated several times to synchronize the receiver [65]. Moreover, DSSS exhibits a threshold behavior to interference. It rejects the interfering signal as long as the interference remains below the jamming margin, but the throughput becomes practically zero if this margin is surpassed [63, 73]. On the other hand, FHSS exhibits a graceful degradation in performance with the increase of interference. Due to this dual behavior, DSSS and FHSS find applications on different domains. The former is typical in the commercial domain (e.g., [31, 32]) where moderate interference levels are caused by users operating on the same spectrum, while the latter finds applications in adversarial settings where the interference is likely caused by a powerful jammer. Because the adversarial model assumed in this work is of a powerful jammer, we develop anti-jamming methods that adopt a FHSS design.

3.1.2 Main Contributions and Chapter Organization

We study the problem of anti-jamming broadcast communications in the presence of inside jammers. We propose the *Time-Delayed Broadcast Scheme* (TDBS) for anti-jamming broadcast communications, based on FHSS communications. TDBS differs from classical FHSS designs in that two communicating nodes do not follow the same FH sequence, but are assigned unique ones that have high correlation properties. Unlike the typical broadcast operation where every receiver is eventually tuned to a common broadcast channel, TDBS implements the broadcast operation as a series of unicast transmissions spread both in frequency and time. To ensure resilience to inside jammers, the locations of the unicast transmissions, defined by a frequency band/slot pair, are only partially known to any subset of receivers. Because the jammer can only interfere with a limited set of frequency bands per time slot, a

subset of the unicast transmissions are interference-free, thus propagating broadcast messages.

The problem of FH sequence design, is mapped to a 1-factorization problem in complete graphs. While a broad class of scheduling algorithms are known to employ 1-factors (perfect matchings) (e.g., [18, 35, 71, 72, 82]), they are, in general, concerned with unicast communications in a hostile setting. They also typically require coordination via the exchange of broadcast messages [18, 35]). TDBS is specifically designed to facilitate broadcasting in the presence of jammers and in the absence of a coordination channel.

Note that TDBS is not meant to be a permanent replacement of the conventional broadcast mechanism in a benign setting. Broadcasting on a common frequency band achieves the optimal communication efficiency (one slot) in the absence of any jammer. TDBS is designed as an emergency mechanism for temporarily restoring communications until the jammer is physically removed from the network. Therefore, its primary focus is resilience to inside jammers and not the communication efficiency of the broadcast operation.

Paper Organization: The remainder of the paper is organized as follows. In Section 3.2, we state the system and adversarial model assumptions. In Section 3.3 we present an overview of TDBS. Section 3.4 describes TDBS for single-hop networks. In Section 3.5, we extend the TDBS operation to multi-hop networks. The security and performance of TDBS are evaluated in Section 3.6. In Section 3.7, we conclude the paper.

3.2 Problem Statement and System Model

3.2.1 Network topologies

We consider two types of network topologies. In the topology of figure 3.1(a), a set of nodes form a single-hop broadcast group. Any node may initiate a broadcast transmission to its neighbors. This single-hop topology is typical in wireless LAN and wireless personal area networks, where a group of devices is assumed to be in

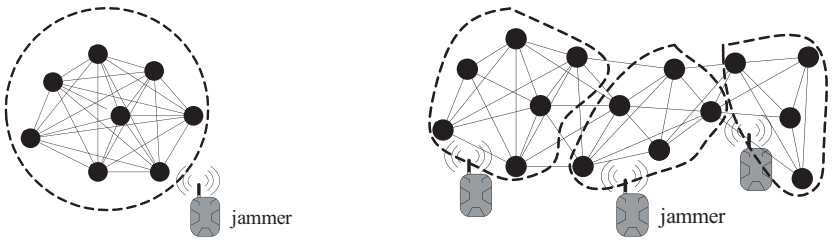


Figure 3.1: (a) A WPAN architecture in which devices located within one-hop form a broadcast communication group, (b) a multi-hop architecture in which communicating nodes span several hops.

close range (e.g., bluetooth devices), and in military scenarios where a set of mobile nodes move in a team-coordinated fashion.

In figure 3.1(b), we consider a multi-hop network connected in ad hoc mode. To make TDBS scalable with the network size, we assume that the network is partitioned to clusters which form cliques [38,83]. Broadcast transmissions occurring under this architecture may be limited within a cluster, or may propagate to other clusters.

3.2.2 System Model

Nodes communicate over a set $\mathcal{C} = \{f_1, \ldots, f_K\}$ of K distinct frequency bands (e.g., $K = 11 - 13$ for 802.11 networks, $K = 79$ for 802.145 (bluetooth)). Each node is equipped with a single half-duplex transceiver. Hence, a node can only listen to or transmit over one band at a time. Note that if communicating devices are assumed to be equipped with more complex hardware such as multiple transceivers, the communication efficiency of TDBS and its resilience to jamming can be improved. We assume that all nodes are synchronized to a time-slotted system. Nodes are capable of hopping between frequency bands. Without loss of generality, we assume that frequency hopping occurs on a per-slot basis, i.e., nodes reside in a frequency band for at least one time slot. For simplicity, the duration of one time slot is assumed sufficient for the transmission of one message unit.

The network is initialized by a trusted authority, which is responsible for pre-

loading relevant parameters such as PN FH sequences and other cryptographic secrets. For multi-hop topologies, we assume a static network topology, known to the trusted authority. Broadcast communications can be either public (transmitted in an unencrypted form) or private. In the latter case, confidentiality and authenticity of the communication is achieved via resource-efficient public key operations. Once the network is initialized, the trusted authority is no longer needed.

3.2.3 Adversary Model

The goal of the adversary is to prevent the sender(s) from communicating with all, or a subset of the intended receivers. For this purpose, the adversary deploys a set of jamming devices at locations of his own choosing, which can be centrally coordinated. These devices are capable of collectively jamming any J frequency bands of the adversary's choosing, by adding interfering signals to the selected frequencies. Wireless transmissions over any of the jammed frequency bands are assumed to be "irrecoverably" corrupted. We do not impose any particular power constraint on the adversary, but assume that the jammed frequency bands become unavailable in the entire network (single-hop, or multi-hop) The jamming devices can switch between frequency bands on a per-slot basis.

The adversary is capable of physically compromising network devices and recovering stored information including cryptographic keys, PN codes, certificates, etc. Moreover, the adversary is aware of the methods used to protect broadcast transmissions (in our case the specifics of the PHY layer implementation and the TDBS algorithm). Note that similar adversary models have been considered in [48,52,54,65].

3.3 Overview of TDBS

To achieve jamming-resistant communications in the presence of insiders, TDBS realizes broadcast as a series of unicast transmissions distributed in frequency and time, thus avoiding the convergence of all nodes to a common frequency band. The locations of the unicast transmissions, defined by a frequency band/slot pair (f, s),

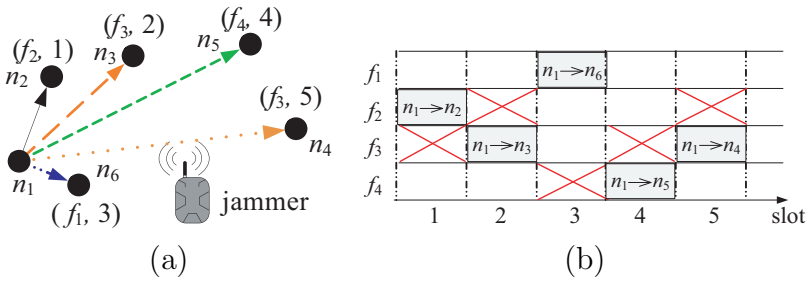


Figure 3.2: (a) Operation in the SU mode. Broadcast is realized as a series of unicasts. The pair (f, s) denotes the frequency band and time slot where the unicast takes place. (b) The timeline of the unicast transmissions of n_1 for the SU mode. The "x" marks denote frequencies jammed by the adversary.

are only partially known to each node (every node is aware of his own schedule). Therefore, the compromise of a node reveals only the set of locations assigned to that node, while keeping the locations of other communicating nodes secret.

For this purpose, nodes are divided into pairs scheduled to communicate over frequency bands which are selected at random. It is possible to partition the set of nodes to groups of size larger than two for more efficient broadcast communication at the expense of reduced resilience to node compromise. Because we are primarily concerned with the jamming-resistance property, we consider the case of node pairs. The communicating pairs and assigned frequency bands change on a per-slot basis thus realizing a FH system. TDBS differs from traditional FH designs in that: (a) communicating nodes do not synchronize to the same FH sequence, but follow unique hopping patterns and, (b) these patterns have a high correlation to lower the number of slots required to complete a broadcast transmission. Moreover, TDBS differs from rendezvous systems that have been proposed for coordinating multi-channel access (e.g. [6,11]), in that it focuses on the broadcast operation as opposed to rendezvous for unicast communications.

Two modes of operation are proposed for TDBS: the sequential unicast mode (SU) and the assisted broadcast mode (AB). In the SU mode, the sender sequentially relays information to intended receivers. This more inefficient mode is appropriate when receivers do not have relaying capabilities, or are not trusted to relay the broadcast message. In the AB mode, any node that receives a broadcast message

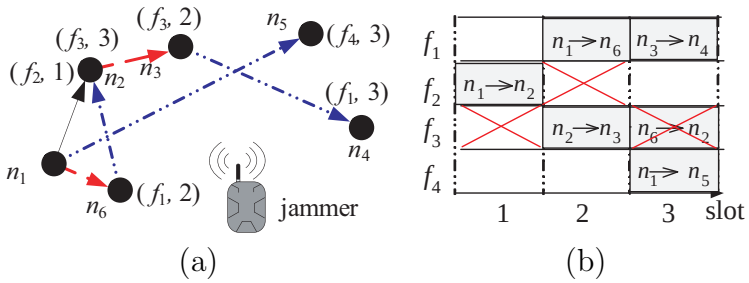


Figure 3.3: (a) Operation in the AB mode. A broadcast transmission is relayed by several nodes at separate frequency bands. (b) The timeline of the unicast transmissions for the AB mode. The "x" marks denote frequencies jammed by the adversary.

can act as a relay for that message.

Figure 3.2 and Figure 3.3 shows an example of the two modes. In figure 3.2(a), node n_1 operates in the SU mode. It sequentially unicasts a broadcast message to nodes $n_2 - n_6$. Figure 3.2(b), depicts the timeline of transmissions of figure 3.2(a). The broadcast is completed after five slots. The "x" marks denote the frequency band jammed by the adversary at each time slot. Figure 3.3(a), shows the operation in the AB mode. Node n_1 initiates a broadcast in slot 1, by transmitting a message m to n_2. In slot 2, n_1 and n_2 relay m to n_6 and n_3, respectively, using frequency bands f_1 and f_3 in parallel. In slot 3, the broadcast is completed with the relay of m from n_1, n_3 and n_6 to n_5, n_4 and n_2, respectively. The timeline of the transmissions taking place in the AB mode is shown in figure 3.3(b). Observe that in this scenario the broadcast is completed despite the jamming of the transmission between n_6 and n_2 in slot 3.

The main challenge of TDBS is to design the FH sequences of individual nodes such that the following requirements are met: (a) hopping sequences are pseudo-random, (b) compromise of a subset of nodes (insiders) limits the information leakage relevant to the sequences of uncompromised nodes, and (c) every node has the same opportunity to perform a broadcast (fairness). In the next section, we develop algorithms for constructing hopping sequences for TDBS-SU and TDBS-AB that satisfy the above requirements. We first illustrate our algorithms for single-hop topologies and then extend our results to multi-hop topologies.

3.4 TDBS for Single-hop Topologies

To achieve resilience to jamming, we randomly distribute unicast transmissions both in frequency and in time. This problem can be viewed as a link scheduling problem for avoiding collisions in multi-channel networks, under the node-exclusive interference model. A large body of literature treats this type of scheduling as various instances of a matching problem in general graphs [18, 35, 71, 72, 82]. However, pre-existing methods are not immediately applicable to our setup for the following reasons.

In link scheduling problems, the goal is to maximize the aggregate network throughput, realized as the sum of individual traffic flows. We are concerned with the dissemination of one message to a specified set of receivers (the members of a broadcast group) over unpredictable frequency band/slot locations, and in the presence of adversaries. This desired property is not necessarily satisfied by maximum throughput designs, which optimally schedule link transmissions in the entire network (centralized approaches) [82]. Moreover, decentralized solutions implementing distributed matching algorithms require the local exchange of coordination messages between nodes, over a commonly agreed channel [18, 35]. Clearly, such a channel cannot be available in our setup due to the presence of an inside jammer.

To ensure the broadcast property, we map the problem of constructing FH sequences to the problem of producing 1-factorizations in complete graphs. 1-factorizations realize a series of perfect matchings (1-factors), which span the all edges of a complete graph [86]. Hence, a broadcast from any node will be communicated to all other nodes. We first present relevant preliminaries from graph theory. Interested readers are referred to [57, 86] for an in-depth treatise of the problem of 1-factorization.

3.4.1 Definitions and Useful Theorems

Consider a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the vertex set and $\mathcal{E}$ denotes the edge set. Assume that $|\mathcal{V}| = 2n$ where n is a positive integer (a dummy node can be

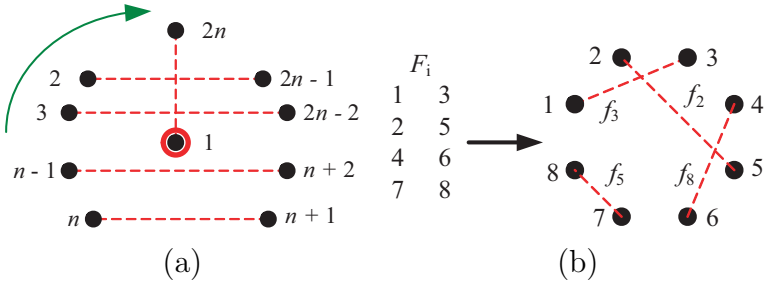


Figure 3.4: (a) Algorithm for constructing a 1-factorization $\mathcal{F} = \{F_0, \ldots, F_{2n-2}\}$. To obtain a factor F_i, every node is rotated by i positions to the left. Node 1 remains fixed. (b) Mapping of a 1-factor to unicast transmissions. Paired nodes concurrently communicate on separate frequency bands.

added otherwise).

Definition 4. Complete Graph: *$\mathcal{G}(\mathcal{V}, \mathcal{E})$ is said to be complete if each pair of vertices is connected by an edge. We denote such a graph by K_{2n}, where $|\mathcal{V}| = 2n$.*

Definition 5. 1-factor: *A 1-factor or a perfect matching F of a graph $\mathcal{G}$ is a subset of $\mathcal{E}$ that partitions $\mathcal{V}$, i.e., F is a set of pairwise disjoint edges of $\mathcal{G}$ that covers all vertices of $\mathcal{V}$.*

Definition 6. 1-factorization: *A 1-factorization $\mathcal{F}_{2n} = \{F_0, F_1, \ldots, F_{2n-2}\}$ of a graph $\mathcal{G}$ is a partition of its edge set $\mathcal{E}$ to $(2n-1)$ 1-factors.*

Theorem 1. 1-factorization of K_{2n}: *A complete graph K_{2n} is 1-factorable [86].*

Construction of 1-factorizations of K_{2n}: 1-factoriza-tions of K_{2n} can be systematically constructed using well-known algorithms (e.g., [27,57,85,86]). These methods typically rely on the selection of a "starter" 1-factor, based on which the entire 1-factorization can be derived. A simple method for constructing a 1-factorization of K_{2n} is to select a starter 1-factor and apply a shift-and-rotate operation to it [86]. This method is illustrated in figure 3.4(a). A 1-factorization is initialized by the 1-factor F_0. Node 1 remains fixed. To obtain the 1-factor F_i, nodes in the perimeter are rotated clockwise by i steps.

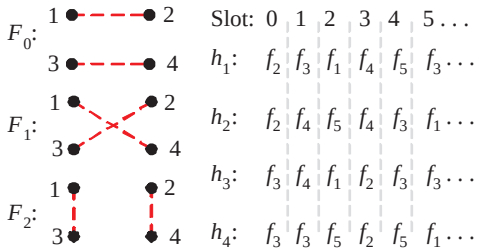


Figure 3.5: Construction of hopping sequences for sequential unicast based on 1-factorization for a group of four nodes.

3.4.2 Mapping to the 1-factorization Problem

In this section, we map the problem of constructing hopping sequences for TDBS into the problem of generating 1-factorizations of complete graphs. In our mapping, the vertex set $\mathcal{V}$ of K_{2n} represents the node set $\mathcal{N}$ of the single-hop network, and an edge $(x, y) \in \mathcal{E}$ represents a unicast transmission between nodes x and y. A 1-factor corresponds to partitioning of the $2n$ nodes into n communicating pairs. These pairs are scheduled to communicate in parallel over separate frequency bands. A 1-factorization $\mathcal{F}_{2n}$ partitions the set of edges $\mathcal{E}$ into $(2n-1)$ disjoint 1-factors, where each edge appears exactly once. In a schedule constructed according to $\mathcal{F}_{2n}$, every node has the opportunity to communicate with all remaining $(2n-1)$ nodes, thus achieving the sequential relay of a broadcast message.

An example of the mapping to the 1-factorization problem is shown in figure 3.4(b). A group of eight nodes is partitioned into four pairs, which are scheduled to communicate over four frequency bands. According to the 1-factor F_i, the communicating pairs during slot i are $\{(1,3),(2,5),(4,6),(7,8)\}$, communicating over frequency bands f_3, f_2, f_8 and f_5, respectively. Figure 3.5 shows a feasible set of hopping sequences h_j for four nodes, $j = 1, \ldots, 4$, based on the 1-factorization of K_4. Communication of all pairs of nodes is completed in three slots. We now present algorithms for constructing FH sequences.

Algorithm 3 TDBS-SU: Sequential Unicast Mode

1: Generate $\mathcal{F}_{2n}$ of K_{2n}
2: **repeat**
3: **for** $i = 0$ to $(2n-2)$ **do**
4: **for** $j = 1$ to $\lceil \frac{n}{K} \rceil$ **do**
5: $\pi = rand(\text{perm}(\mathcal{C}))$
6: **for** $w = 1$ to $\min\{n, K\}$ **do**
7: $h_{F((j-1)K+w,1)} = h_{F((j-1)K+w,2)} = \pi(w)$
8: **end for**
9: **end for**
10: **end for**
11: **end repeat**

3.4.3 TDBS-SU: Sequential Unicast Mode

In the SU mode, a sender sequentially unicasts the broadcast message to $(2n-1)$ intended receivers. The hopping sequences are constructed as follows.

Step 1: Construct a 1-factorization $\mathcal{F}_{2n}$ of K_{2n}, where $\mathcal{F}_{2n} = \{F_0, F_1, \ldots, F_{2n-2}\}$.
Step 2: For all $F_i \in \mathcal{F}_{2n}$, $0 \leq i \leq 2n-2$, repeat Steps 3 to 5.
Step 3: Obtain a random permutation π of the set of frequency bands $\mathcal{C}$.
Step 4: Assign frequency bands in π to $\min\{n, K\}$ edges of F_i in the order of occurrence of the edges.
Step 5: Repeat Steps 3 and 4 until all pairs in F_i are assigned a frequency band.
Step 6: Repeat Steps 1-5.

The pseudo-code of the hopping sequence construction for the SU mode is shown in Algorithm 3. In figure 3.5, we show an example of the application of Algorithm 3 to a group of four nodes. The set of available channels is $\mathcal{C} = \{f_1, \ldots, f_5\}$, $(K = 5)$. Because $K \geq n$, the n pairs corresponding to a 1-factor can communicate in parallel in one slot. In slot 0, pairs communicate according to factor F_0. The random permutation of $\mathcal{C}$ is $\pi = \{f_2, f_3, f_5, f_1, f_4\}$. Pair $(1, 2)$ is assigned band

$\pi(1) = f_2$ and pair $(3, 4)$ is assigned band $\pi(2) = f_3$. The process is repeated for factors F_1, and F_2. Note that condition $K \geq n$ is not necessary for the correct operation of our algorithm. When the number of frequency bands is smaller than the pairs of communicating nodes, transmissions corresponding to one factor are split in multiple slots, as shown in Steps 3-5. However, for single hop networks, it is expected that $K >> n$ (e.g., $K = 79$ in 802.11 and 802.15). We now show that Algorithm 3 constructs random FH sequences.

Proposition 8. *The FH sequences constructed by Algorithm 3 are random.*

Proof. Let $h_j = \{X_1, X_2, \ldots\}$ denote a FH sequence constructed by Algorithm 3 for a node j, where X_i is a random variable denoting the frequency band used at slot i. Random variables X_i form an i.i.d. with each variable being randomly distributed (frequency bands at Step 4 are randomly and independently selected). Hence, h is random. □

3.4.4 TDBS-AB: Assisted Broadcast Mode

Algorithm 4 TDBS-AB: Assisted Broadcast Mode

1: Generate random F_0 of K_{2n}
2: initialize $i = 0$
3: **repeat**
4: **for** $j = 1$ to $\lceil \frac{n}{K} \rceil$ **do**
5: $\pi = rand(\text{perm}(\mathcal{C}))$
6: **for** $w = 1$ to $\min\{n, K\}$ **do**
7: $h_{F_i((j-1)K+w,1)} = h_{F_i((j-1)K+w,2)} = \pi(w)$
8: **end for**
9: **end for**
10: $F_{i+1} = \texttt{split}(F_i)$
11: $i++$
12: **end repeat**

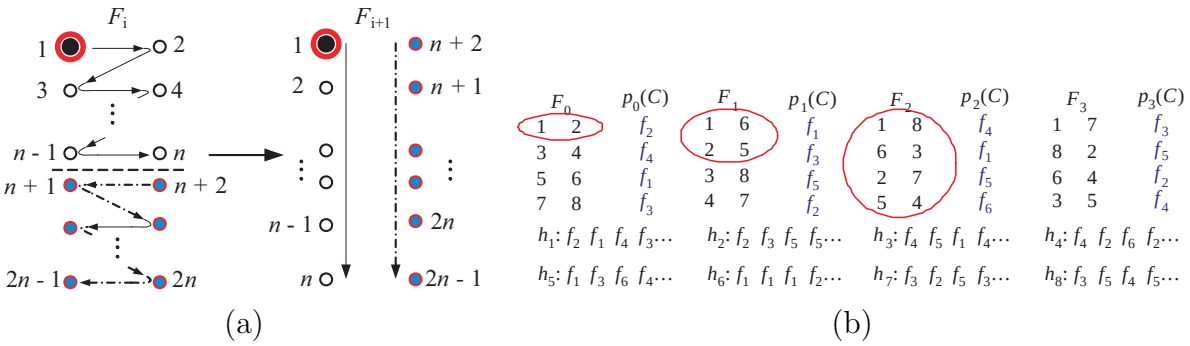


Figure 3.6: (a) Splitting algorithm used to obtain the 1-factor F_{i+1} from the 1-factor F_i. The first n nodes of F_i are obtained in a "zigzag" fashion and are placed on the first column of F_{i+1}. The last n nodes of F_i are obtained in an "inverse zigzag" fashion and are placed in the second column of F_{i+1}. (b) The first four 1-factors for a group of eight nodes and the corresponding hopping sequences.

In the AB mode, any node that has already received a broadcast message operates as a broadcast relay. To construct hopping sequences for the AB mode, the 1-factors F_i are selected and arranged in such a way that the *number of nodes that can relay a broadcast transmission in each 1-factor is maximized*. This property minimizes the delay until the broadcast is completed, while increasing resilience to jamming. We first define the notion of the *relay set*.

Definition 7. The Relay Set R_j^i *of node* j *in a 1-factor* F_i *is defined as the set of nodes that can relay a transmission that originated from* j*.*

The main idea of our hopping sequence construction algorithm is to maximize the size of the relay set R_j^i, for every node j and in every 1-factor F_i. Note that in the AB mode, it is not necessary that the series of 1-factors form a 1-factorization (i.e., that all pairs of nodes communicate directly), because nodes can receive the broadcast transmission via multiple hops. The hopping sequences assigned to each node are constructed as follows.

Step 1: Obtain an arbitrary 1-factor F_0 of K_{2n}. Set $i = 0$.

Step 2: Obtain a random permutation π of the set of frequency bands $\mathcal{C}$.

Step 3: Assign frequency bands in π to $\min\{n, K\}$ edges of F_i in the order of occurrence of the edges.

Algorithm 5 Splitting Algorithm `split`

1: $F_{i+1}(1,1) = F_i(1,1)$
2: **if** n even **then**
3: $\quad F_{i+1}(1,2) = F_i(\frac{n}{2}+1, 2)$
4: **else**
5: $\quad F_{i+1}(1,2) = F_i(\lceil \frac{n}{2} \rceil, 2)$
6: **end if**
7: **for** $j = 2$ to n **do**
8: $\quad F_{i+1}(j,1) = F_i(\lceil \frac{j}{2} \rceil, 2)$, if j even
9: $\quad F_{i+1}(j,1) = F_i(\lceil \frac{j}{2} \rceil, 1)$, if j odd
10: $\quad$ **if** n even **then**
11: $\quad\quad F_{i+1}(j,2) = F_i(\lceil \frac{n+j}{2} \rceil, 1)$, if j even
12: $\quad\quad F_{i+1}(j,2) = F_i(\lceil (\frac{n+j}{2} \rceil, 2)$, if j odd
13: $\quad$ **else**
14: $\quad\quad F_{i+1}(j,2) = F_i(\lceil \frac{n+j}{2} \rceil, 2)$, if j even
15: $\quad\quad F_{i+1}(j,2) = F_i(\lceil (\frac{n+j}{2} \rceil, 1)$, if j odd
16: $\quad$ **end if**
17: **end for**

Step 4: Repeat Steps 2 and 3 until all pairs in F_i are assigned a frequency band.
Step 5: Construct 1-factor F_{i+1} according to the *splitting algorithm*. Set $i = i + 1$.
Step 6: Repeat Steps 2 and 5.

The pseudo-code of TDBS-AB is shown in Algorithm 4. The pseudo-code of the splitting algorithm employed to generate F_{i+1} from F_i is shown in Algorithm 5, and illustrated in figure 3.6(a). Every pair of nodes that communicate according to the 1-factor F_i are placed in adjacent rows in the 1-factor F_{i+1}. The propagation of this property in subsequent 1-factors minimizes the broadcast delay by maximizing the size of the relay set R_j^i for any j and for every 1-factor.

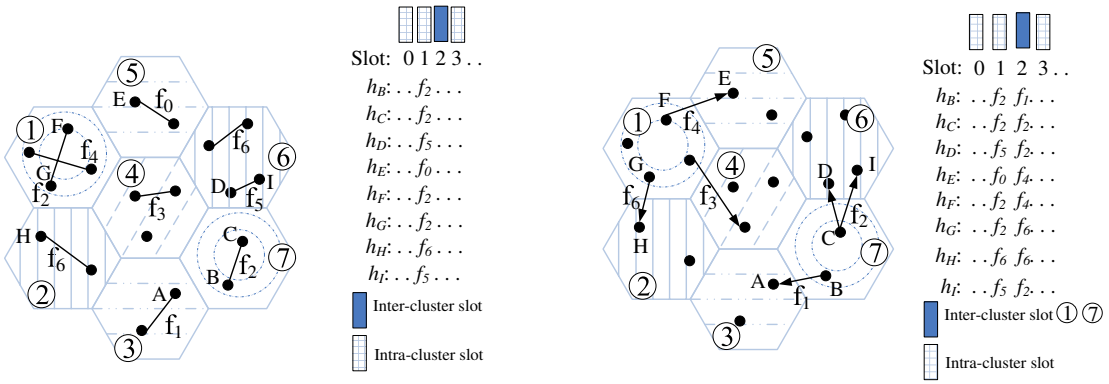
To illustrate the application of Algorithm 4, consider a network of eight nodes. The first four 1-factors that are generated by our algorithm and the corresponding

hopping sequences assigned to various nodes are shown in figure 3.6(b). Node 1 initiates a broadcast transmission of message m following the 1-factor F_0. The circles mark the nodes that receive message m after the completion of the unicasts corresponding to various 1-factors. In fact, one can verify from the 1-factors shown in Fig. 3.6(b) that any broadcast transmission initiated under 1-factor F_0 is completed by 1-factor $F_{\log_2(8)-1} = F_2$. In section 3.6, we prove that this property holds for any broadcast initiated at any time slot. Note that TDBS-AB uses the same mechanism as TDBS-SU (Steps 2-4) for assigning frequency bands to communicating pairs. Therefore, Proposition 8 holds for the hopping sequences generated by TDBS-AB. These sequences are uniformly distributed over the set of available channels, thus minimizing the success of an external jammer in guessing the frequency bands of future communications based on past observations. Moreover, compromise of sequences limits the information leakage regarding other sequences.

3.5 TDBS in Multi-hop Networks

In this section, we extend the operation of TDBS to multi-hop networks. In this scenario, the FH sequence design can be viewed as a global scheduling problem. While several distributed methods have been proposed for distributed scheduling (e.g., [18, 35]), we note that these methods require coordination via a commonly accessible channel. However, such a channel can be blocked by an inside jammer. We, therefore, develop a scalable solution based on clustering, that does not require node coordination.

We partition the network into clusters where each cluster forms a clique [38, 83]. Clique clustering produces a network partition where every node belongs to a single cluster and the members of each cluster are within one hop. We then divide the broadcast operation into two phases: (a) the intra-cluster phase, and (b) the inter-cluster phase. During the intra-cluster phase, communication is limited within each cluster. In the inter-cluster phase, messages are exchanged between border nodes of adjacent clusters. The two phases are interleaved in time.



(a) (b)

Figure 3.7: (a) The intra-cluster phase, (b) the inter-cluster phase.

3.5.1 Intra-cluster Phase

In the intra-cluster phase, a broadcast message propagates to all nodes within a cluster. Because the nodes of a cluster form a clique, the SU or AB operation modes for single-hop networks are employed. To avoid interference between adjacent clusters, the set of available frequency bands $\mathcal{C}$ is divided into four mutually exclusive sets which are assigned to each cluster according to the four color theorem [5].

One such assignment is shown in figure 3.7(a). The shading pattern of each cluster denotes a separate set of frequency bands. In this example, 10 frequency bands are assigned to each cluster, yielding a $K = 40$. Note that the number of available frequency bands K is expected to be be much larger than the number of nodes within the same collision domain (i.e., cluster size). In any case, the algorithms outlined in Sections 3.4.3 and 3.4.4, produce FH sequences for any relation between K and n. The steps for deriving FH sequences for the intra-cluster phase are as follows.

Step 1: Color each cluster based on the four-color theorem.

Step 2: For each distinct cluster size $2n$, construct a 1-factorization $\mathcal{F}_{2n}$ of K_{2n}.

Step 3: For each cluster, pick the 1-factorization corresponding to its cluster size and construct FH sequences for the cluster nodes following the SU mode or the AB mode.

Step 4: Repeat Steps 2 and 3 until all clusters are processed.

In Step 2, it is sufficient to produce distinct 1-factorizations for every possible cluster size. Two clusters of the same size can use the same 1-factorization, dictating the rendezvous of its cluster members, respectively. However, due to the random permutation assignment of frequency bands in Step 3, the pairs of nodes of each cluster will communicate at different frequency bands, thus ensuring the randomness of the pairwise communication among pairs.

3.5.2 Inter-cluster Phase

In the inter-cluster phase, border nodes in adjacent clusters relay broadcast messages that are intended to propagate beyond the boundaries of each cluster. To do so, while avoiding collisions between adjacent transmissions, we exploit the cluster labeling produced by the application of the four-color theorem. During this phase, every time-slot is marked with one of the four colors indicating the set of clusters that are allowed to transmit on that slot. As an example, in figure 3.7, clusters A and D are allowed to transmit on slot 0, clusters C and F on slot 1, clusters B and E on slot 2 and cluster G on slot 3, with this sequence repeating modulo four (slot numbers indicate assignment before the interleaving with the intra-cluster phase) . After the slot coloring, the FH sequences of individual nodes are generated as follows.

Step 1: For each cluster x, find the nodes in x bordering adjacent clusters. Place this nodes to a set $\mathcal{A}$.

Step 2: For each node $i \in \mathcal{A}$, find the neighbors of i in adjacent clusters. If a neighbor is common to two nodes in x, assign it to the node with the fewer neighbors. Break ties arbitrarily (e.g., considering the node with the lowest id). Merge nodes assigned to the same i to a single vertex and place vertices to set $\mathcal{B}$. Create a bipartite graph $G(\mathcal{A} \cup \mathcal{B}, \mathrm{E})$, where an edge (x,y) exists if nodes corresponding to y are assigned to x. G forms a 1-factor F_x.

Step 3: For each slot colored with x's color, obtain a random permutation π of the set of frequency bands $\mathcal{C}$.

Step 4: Assign frequency bands in π to $\min\{n, K\}$ edges of F_x in the order of occurrence of the edges.
Step 5: Repeat Steps 3 and 4 until all pairs in F_x are assigned a frequency band.
Step 6: Repeat Steps 1-5, until all clusters are processed.

The inter-cluster phase is illustrated in Figure 3.7(b). According to their color, clusters A and D are scheduled to broadcast during slot 0. Nodes 2,3, and 4 belong to set $\mathcal{A}$ of cluster A since they can communicate with nodes of adjacent clusters. For slot 0, the communicating pairs are $\{2-9, 10\}$ $\{3-11, 12\}$ and $\{4-7, 8\}$, and are assigned frequency bands f_{11}, f_{22}, and f_2, respectively. Similarly, for cluster D and slot 0, the communicating pairs are $\{5-6, 13\}$ $\{14-15\}$ and $\{16-17\}$, and are assigned frequency bands f_8, f_{33}, and f_{25}, respectively. Note that during the inter-cluster phase, all channels in $\mathcal{C}$ are available for assignment to the communications of adjacent pairs of nodes.

The intra-cluster and inter-cluster slots are interleaved in the FH design, to allow for both single hop and multi-hop broadcast transmissions are achieved.

3.6 Performance and Security Evaluation

In this section, we evaluate the performance of TDBS and analyze its security properties. As a performance/security metric, we use the broadcast delay, defined as follows.

Definition 8. The Broadcast Delay *D is the number of slots required for the completion of a broadcast operation, i.e., until all intended recipients have received a copy of the broadcasted message.*

3.6.1 Performance in the Absence of Jammers

In this section, we evaluate the broadcast delay for the two TDBS modes in the absence of jammers. This analysis is provided to facilitate the evaluation of the broadcast delay when jammers are assumed to be present.

Proposition 9. *The broadcast delay of TDBS-SU is* $D = \lceil \frac{n}{K} \rceil (2n-1)$ *slots.*

Proof. To complete a broadcast in the SU mode, the sender must unicast the broadcast message to $2n-1$ receivers. The $2n-1$ transmissions correspond to the $(2n-1)$ 1-factors of $\mathcal{F}_{2n}$ (each node is scheduled to communicate once per 1-factor). Each factor requires $\lceil \frac{n}{K} \rceil$ time slots to be completed (here, all transmissions of a 1-factor are completed before transmissions of other 1-factors can proceed, in order to avoid schedule conflicts). Hence, the broadcast delay is equal to $\lceil \frac{n}{K} \rceil$ times the number of factors of $\mathcal{F}_{2n}$. □

Next, we evaluate the broadcast delay in the AB mode.

Proposition 10. *The broadcast delay for TDBS-AB is* $D = \lceil \frac{n}{K} \rceil \lceil \log_2(2n) \rceil$ *slots.*

Proof. We first prove that any broadcast transmission in the AB mode is completed after $\lceil \log_2(2n) \rceil$ 1-factors. Without loss of generality, assume that a broadcast is initiated by node $F_i(k,1)$, located in the kth row of F_i. With the completion of F_i, the relay set is $R_i^j = \{F_i(k,1), F_i(k,2)\}$. After the execution of Algorithm 5, nodes $F_i(k,1)$ and $F_i(k,2)$ appear in adjacent rows (due to the cyclic nature of Algorithm 5, rows 1 and 8 are considered to be adjacent) on the 1-factor F_{i+1}. This can be easily verified by reversing the mapping from F_{i+1} to F_i in lines 8-15 of Algorithm 5. Because the pair $(F_i(k,1), F_i(k,2))$ appears on separate rows on F_{i+1}, each node will relay the broadcast message to two new nodes, thus increasing R_{i+1}^j to four nodes.

Further execution of Algorithm 5 divides the nodes in the relay set R_{i+1}^j to four adjacent rows. Since none of the nodes in R_{i+1}^j appears on the same row, the relay set after the completion of factor F_{i+1} increases to eight nodes. Following the recursive application of the splitting algorithm, the relay set after the completion of $\lfloor \log_2(2n) \rfloor$ 1-factors has a size of $2^{\lfloor \log_2 2n \rfloor}$. If $\lfloor \log_2(2n) \rfloor = \log_2(2n)$, the broadcast is complete since $2^{\log_2(2n)} = 2n$. Otherwise, one extra 1-factor is needed to relay the broadcast to the remaining $2n - 2^{\lfloor \log_2(2n) \rfloor}$ nodes. Because $2^{\lfloor \log_2(2n) \rfloor} > n$, the splitting algorithm places n nodes from the relay set into the n rows of the $\lfloor \log_2 2n \rfloor + 1 = \lceil \log_2(2n) \rceil$th 1-factor. These n relays complete the broadcast operation. Combining the two

cases yields a required number of 1-factors that is equal to $\lceil \log_2(2n) \rceil$. Proposition 10 follows by noting that every 1-factor requires $\lceil \frac{n}{k} \rceil$ slots to be completed. □

3.6.2 Security Analysis

We first analyze the resilience of TDBS to external and internal jammers for single-hop networks.

Resilience to External Jammers

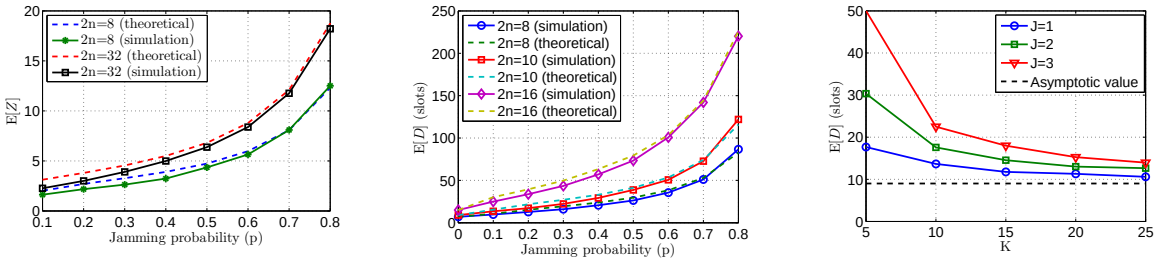
Under an external threat model, the hopping sequences assigned to various nodes remain secret. For this scenario, we assume that the adversary deploys multiple jamming devices that can jam up to J frequency bands per time slot, with $J < K$. For convenience, we assume $K \geq n$ so that all node pairs corresponding to a 1-factor can communicate in parallel in one time-slot. This is typical in wideband communications where K is much larger than the expected number of nodes within the same collision domain. Our results can be extended to the $K < n$ case in a straightforward manner. Suppose that a jammer attempts to jam the broadcast of a single node j. To compute D, we evaluate the average number of 1-factorizations needed to complete the broadcast, in the presence of the external jammer, and for each mode.

Proposition 11. *In the presence of an external jammer, the expected number* $\mathrm{E}[Z]$ *of 1-factorizations needed to complete a broadcast operation in the SU mode is*

$$\begin{aligned} \mathrm{E}[Z] &= (1-p)^{2n-1} + \sum_{i=2}^{\infty} i(1-p^{i-1})^{2n-1} \times \\ &\quad \sum_{k=1}^{2n-1} \binom{2n-1}{k} \left(\frac{p^{i-1}(1-p)}{1-p^{i-1}} \right)^k, \end{aligned} \tag{3.1}$$

where $p = \frac{J}{K}$ *denotes the jamming probability.*

Proof. Suppose that an arbitrary node j attempts a broadcast transmission in the presence of an external jammer. This broadcast is completed in a single 1-factorization if the jammer is unsuccessful in jamming the communication of j for



(a) (b) (c)

Figure 3.8: (a) E[Z] as function of the jamming probability p, (b) E[D] as a function of jamming probability p. (c) E[D] as a function of K when $2n = 10$.

$2n-1$ consecutive slots. Because h_j is random, a transmission of node j is successful with probability $\left(1 - \frac{J}{K}\right)$. Moreover, the events of a successful transmission of node j at slot i and slot w, $i \neq w$ are independent. Hence,

$$\Pr[Z = 1] = \left(1 - \frac{J}{K}\right)^{2n-1} = (1-p)^{2n-1}.$$

The broadcast is completed in two 1-factorizations if every receiver is jammed at most one time, and at least one receiver is jammed on the first 1-factorization. Taking into account all possible combinations,

$$\Pr[Z = 2] = \sum_{k=1}^{2n-1} \binom{2n-1}{k} (1-p)^{2n-1-k} p^k (1-p)^k.$$

Generalizing to the case of $Z = i$, it follows that

$$\begin{aligned}\Pr[Z = i] &= \sum_{k=1}^{2n-1} \binom{2n-1}{k} (1-p^{i-1})^{2n-1-k} \\ &\quad p^{(i-1)k}(1-p)^k, \\ &= (1-p^{i-1})^{2n-1} \sum_{k=1}^{2n-1} \binom{2n-1}{k} \\ &\quad \left(\frac{p^{i-1}(1-p)}{1-p^{i-1}}\right)^k.\end{aligned}$$

Proposition 11 follows from the definition of the expectation, i.e., E[Z] = $\sum_i i \Pr[Z = i]$. □

In figure 3.8(a), we compare the theoretical value of E[Z] with the simulated one. For our simulations, we generated sequences of size $1,000$ hops for different values of n and K according to Algorithm 3. We also randomly selected J channels to be jammed per time-slot. All results were averaged over 100 runs. We measured E[Z] as a function of the jamming probability $p = \frac{J}{K}$. We observe that the simulation values agree with the theoretical ones.

Based on Proposition 11, the expected broadcast delay E[D] is equal to the expected number of 1-factorizations needed for the completion of a broadcast, times the number of slots needed for the completion of each 1-factorization. The first $(E[Z] - 1)$ 1-factorizations require $(2n - 1)$ slots, while the last 1-factorization requires, on average, $\frac{2n-1}{2}$ slots (the last successful transmission takes place on any of the 1-factors of the last 1-factorization with equal probability). Therefore, $\mathrm{E}[D] = (2n - 1)\left(\mathrm{E}[Z] - \frac{1}{2}\right)$.

Figure 3.8(b), shows the theoretical and simulated value of E[D] as a function of the jamming probability p. We observe that even when the adversary jams 80% of the available channels, nodes are still capable of completing their broadcast transmissions at the expense of some delay. Nevertheless, the broadcast communication is maintained. In figure 3.8(c), we show E[D] as a function of the number of available channels K for various values of J. E[D] decreases with K, approaching the asymptotic value of K, obtained in the absence of a jammer, i.e., $\mathrm{E}[D] = 2n - 1$.

For the AB mode, E[D] does not have a simple closed-form expression but involves complex summation formulas. However, we can derive a useful formula for $J = 1$.

Proposition 12. *After the first successful relay of a broadcast message m, the broadcast delay until m is received by $(2n-2)$ nodes (all nodes, but one) is bounded by*

$$\lceil \log_2(2n) \rceil - 1 \leq D \leq \lceil \log_2(2n) \rceil. \tag{3.2}$$

Proof. The lower bound immediately follows from Proposition 10. The broadcast delay in the absence of a jammer is equivalent to the delay in the presence of

an external jammer who is unsuccessful in jamming any communicating pair for $\lceil \log_2(2n) \rceil - 1$ slots. Hence, after the first successful relay, the lower bound on D follows.

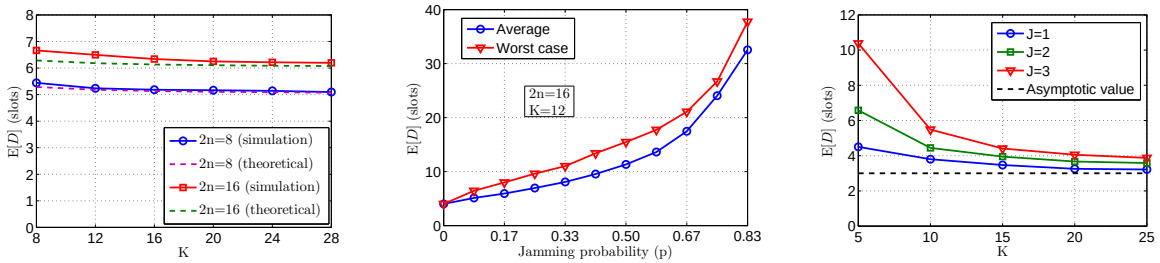
To compute the upper bound on D, assume that an arbitrary node j wants to broadcast a message m to the remaining $(2n-1)$ nodes. Let a_i denote the size of the relay set in slot i. Initially, $a_0 = 2$, i.e., node j has completed its first successful relay. Once $a_i \geq 2$, the adversary can jam at most one of the pairs relaying m. The size of the relay set in this worst-case scenario grows according to the formula.

$$a_i = 2a_{i-1} - 1 = 2^i + 1, \ i \leq \lceil \log_2(2n) \rceil - 1, \tag{3.3}$$

where a_i is computed recursively with $a_0 = 2$. To show the validity of (3.3), we refer to the proof of Proposition 10, where we showed that for $a_i \leq n$, the size of the relay set doubles with the increment of i. Because the adversary jams at most one frequency band per time slot, in the worst case, $a_i = 2a_{i-1} - 1$. This is true until $a_i \geq n$, in which case the size of the relay set can no longer double. In slot i, $i \leq \lceil \log_2(2n) \rceil - 1$, the relay set becomes larger than n for the first time. That is, it takes $i = \lceil \log_2(2n) \rceil - 1$ slots until more than half the nodes can relay message m. These $a_i \geq n$ relay nodes communicate with the remaining $2n - 2^i - 1 \leq n$ nodes that have not yet received m. Since the adversary can only jam one frequency band, the number of nodes that have received m at the end of slot $(i+1)$ is equal to $(2n-2)$. That is, in the worst case, only one node has not received m after $\lceil \log_2(2n) \rceil$ slots. □

Proposition 12 allows us to estimate the expected broadcast delay for the AB mode. Let D_1 denote the expected delay until the first success, D_2 the delay until $(2n-2)$ nodes receive message m and D_3 the delay until the last node receives m. The expected broadcast delay is bounded by

$$\begin{aligned} \mathrm{E}[D] &= \mathrm{E}[D_1 + D_2 + D_3] \\ &\leq \frac{K}{K-1} + \lceil \log_2(2n) \rceil + \frac{K}{K-1}. \end{aligned} \tag{3.4}$$



(a) (b) (c)

Figure 3.9: (a) E$[D]$ as a function of K when $J = 1$, for the AB mode of the worst case. The theoretical value is computed based on (3.4). (b) E$[D]$ as a function of p, for the AB mode. The average and worst case are shown. (c) E$[D]$ as a function of K and for various J. The asymptotic value is equal to $\lceil \log_2(2n) \rceil$.

In (3.4), we have used the fact that it takes, on average, $\frac{K}{K-1}$ slots for the first successful relay when $p = \frac{1}{K}$. Moreover, after the first success, $\lceil \log_2(2n) \rceil$ slots are needed in the worst case until $2n - 2$ nodes receive m. The last node receives m after $\frac{K}{K-1}$ slots, on average.

We also studied the performance of the AB mode via simulations. For our simulations, we generated sequences of size $1,000$ hops for different values of n and K according to Algorithm 4. We also randomly selected J channels to be jammed per time-slot. All results were averaged over 100 runs. Figure 3.9(a) shows the value of E$[D]$ as a function of K for $J = 1$. We observe that the theoretical value derived using Proposition 12 agrees with the value obtained via simulation. In figure 3.9(b), we show the average and worst-case broadcast delay, as a function of the jamming probability p. We observe that even when p is as high as 0.83, the average and worst-case delays differ by less than six slots. This is due to the "relay explosion" effect of the splitting algorithm.

The AB mode is significantly more resilient to jamming than the SU mode, due to the larger number of nodes relaying the broadcast message. Even when 83% of the frequency bands are jammed, the AB mode requires only 38 slots to complete a broadcast, compared to 228 slots needed with the SU mode. In figure 3.9(c), we show E$[D]$ as a function K for different values of J. We observe that with the increase of K, E$[D]$ asymptotically approaches the performance of the AB mode in

the absence of jammers.

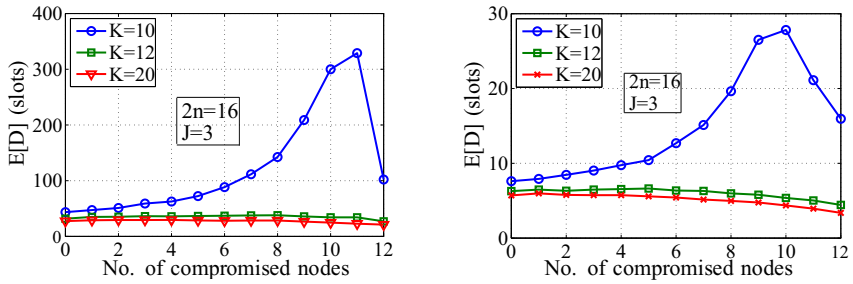
Resilience to Internal Jammers

Assume now that the adversary has compromised r nodes and recovered their FH sequences. We are interested in determining the broadcast delay until the remaining $(2n - r - 2)$ *legitimate* nodes receive a broadcast message m from a legitimate source. Knowledge of the r hopping sequences reduces the adversary's uncertainty with respect to the frequency bands where unicast transmissions between legitimate nodes take place. This is because at a given time slot, communications take place in orthogonal frequency bands. Thus knowing r FH sequences reduces the space of $\mathcal{C}$ for the selection of the uncompromised FH sequences. The exact value of E[D] depends on the selection of the 1-factorization that is used to construct the hopping sequences and the specific arrangement of the compromised nodes on that 1-factorization. The jamming probability p varies on a slot-by-slot basis and is given in the following proposition.

Proposition 13. *Under the compromise of r nodes, the jamming probability p is bounded by*

$$\min\{1, \frac{J}{K - \lceil \frac{r}{2} \rceil}\} \leq p \leq \min\{1, \frac{J}{K - r}\}. \tag{3.5}$$

Proof. Let x be the number of frequency bands over which the r compromised nodes are scheduled to communicate according to the 1-factor F. The number of bands over which legitimate communications take place in each slot is reduced to $K - x$. Hence, the jamming probability is increased to $p = \frac{J}{K-x}$. To derive bounds on p, we consider the lowest and highest values of x. If the compromised nodes are scheduled to communicate with each other at 1-factor F, then $x = x_{\min} = \lceil \frac{r}{2} \rceil$, where the ceiling function is used to account for an odd value of r. This value of x yields the lower bound on p. On the other hand, if all r nodes are scheduled to communicate with legitimate nodes (appear on separate rows in F), then $x = x_{\max} = r$. In this case, p attains its maximum value. Note that $p \leq 1$ and hence, $r \leq K - J$. When



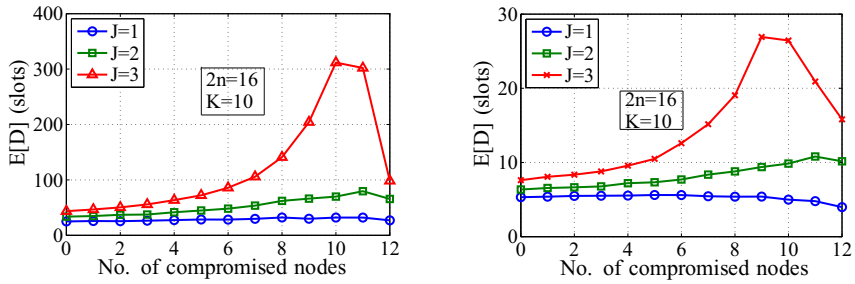
(a) SU mode (b) AB mode

Figure 3.10: E[D] as a function of the number of compromised nodes for various values of K, when $J = 3$.

r is larger than $K - J$, there are 1-factors where all legitimate transmissions are jammed with certainty. □

We further used simulation to investigate the impact of node compromise on the broadcast delay. For our simulations, we generated FH sequences of size $1,000$ hops for different values of n and K. We randomly selected r of these sequences to be exposed to the adversary. At each time slot, the adversary randomly selected J frequency bands to be jammed, excluding the ones already known due to node compromise. A broadcast was assumed to be completed, when all legitimate nodes were able to obtain a broadcast message. All results were averaged over 100 runs. Figures. 3.10(a) and 3.10(b) show E[D] as a function of the number of compromised nodes when $J = 3$ and $K = 10, 12, 20$, for the SU and AB modes, respectively. We observe that legitimate nodes are able to complete their broadcast transmissions even when more than 50% of the nodes are compromised. The AB mode exhibits significantly lower delay compared to the SU mode, due to the use of multiple nodes as relays. Note that when K is small and several nodes are compromised, the jammers have very high chance to hit legitimate pairs. This fact can be seen from the sharp increase of E[D] when $K = 10$ and more than half of the nodes are compromised. A slightly increase of K will alleviate the situation greatly.

In figure 3.11(a) and 3.11(b), we show E[D] as a function of the number of compromised nodes when $K = 10$ and for various values of J, under the SU and AB modes, respectively. Even with the increase of J, legitimate nodes are able to



(a) SU mode (b) AB mode

Figure 3.11: E[D] as a function of the number of compromised nodes for various values of J, when $K = 10$.

complete their broadcast transmissions in both modes, with the AB mode being the most efficient. Note that E[D] starts to decrease when a large number of nodes is compromised. This is due to the fact that with the compromise of more nodes, fewer legitimate nodes need to receive a broadcast transmission, thus reducing the number of unicast transmissions that need to be completed.

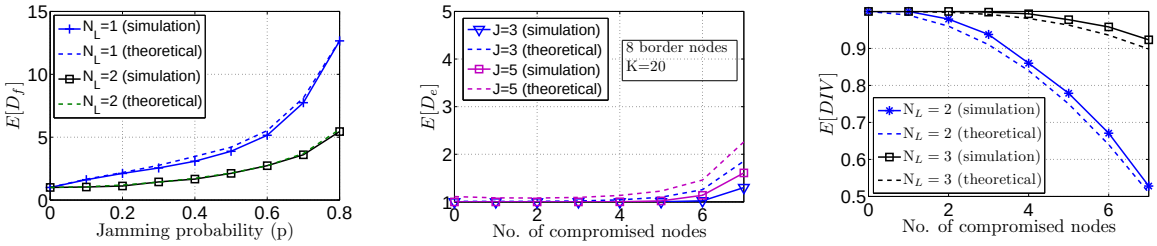
3.6.3 Evaluation of Multi-hop Scenarios

In this section, we evaluate TDBS for multi-hop networks. We focus on the jamming-resistance of the inter-cluster phase, since for the intra-cluster phase, the security analysis for single-hop networks holds. We define the following performance metrics for the inter-cluster phase:

- **Flooding Delay** D_f**:** the number of slots needed until all clusters adjacent to a cluster i, have received a broadcast that originated in i, directly from a node in i.

- **Escape Delay** D_e**:** the number of slots needed until a broadcast message m originating at a cluster i, reaches any node in any adjacent cluster.

Escape diversity DIV**:** the fraction of adjacent clusters that receive the broadcast message m directly from a cluster i, when some border nodes in i are compromised.



(a) (b) (c)

Figure 3.12: (a) E$[D_f]$ as a function of the jamming probability p, (b) E$[D_e]$ as a function of the number of compromised nodes r for various J, (c) E$[DIV]$ as a function of the number of compromised nodes r for various N_L.

We first analytically evaluate the average flooding delay E$[D_f]$ in the presence of external jammers. Assume a cluster with N_C adjacent clusters. Let N_L denote the number of "bridge links" between two adjacent clusters.

Proposition 14. *In the presence of an external jammer,* E$[D_f]$ *is equal to*

$$\begin{aligned} \mathrm{E}[D_f] &= (1-\tilde{p})^{N_C} + \sum_{i=2}^{\infty} i(1-\tilde{p}^{i-1})^{N_C} \times \\ &\sum_{k=1}^{N_C} \binom{N_C}{k} \left(\frac{\tilde{p}^{i-1}(1-\tilde{p})}{1-\tilde{p}^{i-1}}\right)^k, \end{aligned} \tag{3.6}$$

where $\tilde{p} = \left(\frac{J}{K}\right)^{N_L}$ *denotes the probability that all* N_L *links to an adjacent cluster are jammed at a particular slot*

Proof. The proof of Proposition 14 follows the same steps as the proof of Proposition 11, by substituting $p = \frac{J}{K}$ with $\tilde{p} = \left(\frac{J}{K}\right)^{N_L}$. Due to space limitations we refer to the proof provided in Proposition 11. □

We also verified Proposition 14 via simulations. In our setup, we generated a multi-hop topology consisting of 50 nodes, organized in clusters. We then generated FH schedules for all nodes in the network for the inter-cluster phase, according to the algorithm described in Section 3.5.2. At each time slot, the jammer was assumed able to block J random frequency bands across the entire network. Results were averaged over all clusters in the network.

Figure 3.12(a) shows $\mathrm{E}[D_f]$ as a function of the jamming probability p. We denote the number of "bridge links" between two adjacent clusters to be N_L. We observe that, even when 80% of the available frequency bands are jammed, only 13 inter-cluster slots are needed for all neighboring clusters to directly receive a broadcast message. Once the message is received by nodes in adjacent clusters, it can propagate within adjacent clusters during the intra-cluster phase.

We also evaluate the expected escape delay $\mathrm{E}[D_e]$ under the compromise of r border nodes.

Proposition 15. *Under the compromise of r border nodes of a cluster i, $\mathrm{E}[D_e]$ is given by*

$$\mathrm{E}[D_e] = \frac{1}{1 - \left(P_c^{N_L} + \sum_{i=1}^{N_L} \binom{N_L}{i} \left(\frac{J(1-P_c)}{K-r} \right)^i \right)^{N_C}}, \tag{3.7}$$

where $P_c = \frac{r}{N_C \times N_L}$ denotes the compromise probability.

Proof. At each time slot, the probability that an adjacent cluster fails to receive a broadcast is due to: (a) all N_L links are shared with compromised border nodes, and (b) the links shared with uncompromised border nodes are jammed by the adversary. So the probability that a neighboring cluster fails to receive a broadcast is

$$P_{fail} = P_c^{N_L} + \sum_{i=1}^{N_L} \binom{N_L}{i} \left(\frac{J(1-P_c)}{K-r} \right)^i.$$

The probability that at least one of the neighboring clusters successfully receive the broadcast at a time slot is

$$P_{success} = 1 - P_{fail}^{N_C}.$$

The broadcast among adjacent nodes forms a Bernoulli trial with a success probability $P_{success}$, so the average delay until the first success is $1/P_{success}$, which leads to our result. □

The expected escape diversity $\mathrm{E}[DIV]$ is evaluated in the following proposition.

Proposition 16. *Under the compromise of r nodes, $\mathrm{E}[DIV]$ is given by*

$$\mathrm{E}[DIV] = 1 - P_c^{N_L}. \tag{3.8}$$

Proof. For any neighboring cluster, the probability that it can not receive a broadcast is that all N_L links are shared with compromised border nodes. This probability is $P_c^{N_L}$. So the expected number of neighboring clusters that can get a broadcast is $N_C \cdot (1 - P_c^{N_L})$. Dividing this value with N_C, yields E$[DIV]$. □

Figures 3.12(b) and 3.12(c) evaluate E$[D_e]$ and E$[DIV]$ as a function of the number of compromised border nodes. In our setup, compromised border nodes do not assist in the broadcast relay and their FH sequences are revealed to the adversary. From figure 3.12(b), we observe that even when most of the border nodes are compromised, a small number of slots is sufficient for the first copy of a broadcast message to reach one adjacent cluster. From figure 3.12(c), we observe that more than 90% of neighboring clusters are guaranteed to receive the message when $N_L = 3$, while this value being reduced to 50% when $N_L = 2$.

3.7 Conclusions

We proposed TDBS, a scheme for jamming-resistant broadcast communications in the presence of inside jammers. In TDBS, broadcast is realized as a series of unicast transmissions distributed in frequency and time. Because the adversary is limited in the number of channels he can jam, several unicast transmissions remain interference-free. We mapped the problem of constructing hopping sequences for TDBS to the problem of 1-factorization of complete graphs. We analytically evaluated the security properties of TDBS under an external and an internal threat model and showed that TDBS maintains broadcast communications even when multiple nodes are compromised. We verified our theoretical analysis using extensive simulations.

CHAPTER 4

Spectrum Opportunity-Based Control Channel Assignment in Cognitive Radio Networks

4.1 Introduction

4.1.1 Motivation

Fixed spectrum allocation policies address interference between different wireless technologies by isolating their operation in frequency. This fixed allocation policy has led to spectrum scarcity. As a consequence, new wireless services and technologies have strived to co-exist in overcrowded unlicensed bands with poor radio propagation characteristics. At the same time, it has been lately observed that portions of the licensed spectrum are highly underutilized [3, 28].

To address the emerging need for higher spectral efficiency, an alternative policy that allows unlicensed users to opportunistically access vacant portions of the licensed spectrum is currently being examined [3,28]. In this so-called opportunistic spectrum access (OSA) paradigm, users are classified as primary if they are licensed to operate on a particular frequency band, and secondary otherwise. Secondary users can operate in licensed frequency bands only if the do not interfere with primary radio (PR) activity. Cognitive Radios (CRs) are intended as an enabling technology for OSA [3]. These devices are capable of sensing the spectrum for frequency holes and adapting their radio parameters to exploit spectrum opportunities without interfering with PRs.

Establishing a self-organizing cognitive radio network (CRN) requires extensive exchange of control messages, needed to coordinate various network functions such as cooperative sensing, channel access, topology management, and routing, to name a few. In many wireless networks architectures, control messages are broadcasted

over a channel known to all nodes, commonly referred to as *the control channel*. This channel can be realized in a number of ways. For example, in multi-channel system, it can be a frequency band dedicated to control traffic (e.g., [76]). It can also be a designated time slot in a TDMA system, or a frequency hopping sequence (or CDMA code) in spread spectrum systems. In an opportunistic CRN, spectrum availability exhibits the temporal and spatial variations due to PR activity. Therefore, there is no guarantee that a given frequency band will be available for exchanging control information, either locally (one-hop neighborhood) or over multiple hops [94]. We refer to the problem of defining a channel for control purposes as the *control-channel assignment (CCA)* problem.

One possible solution to the CCA problem is to license a slice of the spectrum for control purposes [15]. However, such a design conflicts with the opportunistic nature of CRNs. Alternatively, the control channel can reside within an unlicensed band, such as the Industrial, Scientific and Medical (ISM) band, or the unlicensed ultra-wide band (UWB) [13, 15]. The use of unlicensed bands jeopardizes the reliability of the control channel, given that such bands are already overcrowded and can experience uncontrollable interference from other unlicensed users. In the absence of a dedicated frequency band, control traffic has to be carried in-band. In such a case, the control channel is subject to PR dynamics, and hence its allocation has to vary in frequency and time according to the locally perceived spectrum availability [7,48,94].

4.1.2 Main Contributions and Chapter Organization

We develop cluster-based methods for CCA in CRNs. This is an intuitive approach given the inherent partitioning of the network into clusters due to the location- and time-dependent spectrum availability. We formulate the clustering problem as a *bipartite graph problem*. In particular, we map the clustering process to the maximum edge biclique problem [24,61] and the maximum one-sided edge cardinality problem [24]. Our mapping allows us to control the tradeoff between the set of common idle channels within each cluster and the cluster size.

Based on our graph theoretic formulations, we develop a distributed cluster-

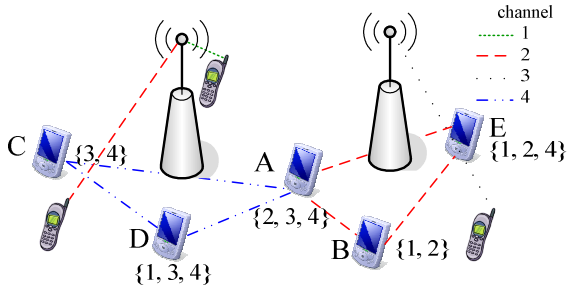


Figure 4.1: Control channel assignment based on PR activity (idle frequency channels are indicated between braces).

ing algorithm called Spectrum Opportunity-based Clustering (SOC). SOC clusters neighboring CRs with similar channel availability. We show that such a criterion reduces the frequency of reclustering due to PR dynamics. Under SOC, nodes reach a mutual agreement with respect to cluster memberships after the local exchange of a small number of messages. Finally, to account for PR dynamics in the time domain and to enable inter-cluster coordination, we propose a periodic channel rotation mechanism. By rotating the control channel among the list of common idle channels in the cluster, nodes in that cluster can communicate with a neighboring cluster as long as the two clusters have at least one common idle channel.

The remainder of this paper is organized as follows: we formalize the CCA problem and present our system model in Section 4.2. Section 4.3 shows the mapping of the CCA problem to a bipartite graph problem. In Section 4.4, we describe the spectrum opportunity clustering (SOC) algorithm. A CCA mechanism that adapts to the spectrum dynamics is presented in Section 4.5. In Section 4.6, we develop a coordination protocol for CRNs for the initial exchange of information before a control channel is established. In Section 4.7, we evaluate the performance of SOC and compare it with other methods. In Section 4.8, we summarize our contributions.

4.2 Problem Statement and System Model

Problem Statement–The CCA problem addressed in this paper is illustrated in Figure 4.1, where a cellular network that acts as a PRN co-exists with an ad hoc network of CRs. CRs opportunistically use any of the four cellular channels that is

sensed idle in its vicinity. For example, the idle channel list of CR node A, denoted as CR_A consists of channels $\{2, 3, 4\}$. Our goal is to allow CRs to agree on a CCA according to their spectrum vacancies. In the example in Figure 4.1, none of the idle channels is common to all CRs. Hence, different channels have to be assigned for control in different neighborhoods. This assignment leads to a natural partitioning of the CRN into clusters, each with at least one common idle channel. To assign the control channel, we investigate clustering algorithms that take into account the spatial and temporal PR activity dynamics.

System Model–We consider a CRN that co-exists with one or more PRNs in the same geographical area. PRs are licensed to operate on a fixed spectrum, which can be divided into a set of M *non-overlapping* frequency bands. Let $\mathcal{M} = \{1, 2, \ldots, M\}$ be such a set. For simplicity, it is assumed that these bands are of equal capacity, and that the CRN maintains the channelization structure of the PRNs. We also assume that CRs have the same communication range over every channel in $\mathcal{M}$. Hence, the connectivity graph is not impacted by which channel is used for control. This property can be achieved, for example, by adjusting the transmission power.

CRs sense spectrum opportunities using energy detectors, cyclostationary feature extraction, or pilot signals. They may exchange their sensed data and cooperate in identifying spectrum opportunities [3, 33, 34, 58]. Spectrum sensing is conducted at a sufficient rate such that the list of available channels at each CR is up-to-date. However, the sensing process is assumed to be imperfect due to multipath fading and/or severe shadowing [33, 34, 58]. Such phenomena are typical in the bands that are likely to be open for CRN use (e.g., digital TV bands [29, 58]).

For the ith CR denoted by CR_i, its list of idle channels is denoted by $C_i = \{f_1^{(i)}, f_2^{(i)}, \ldots, f_{K_i}^{(i)}\}$, where $K_i = |C_i|$. Here, $f_j^{(i)}$ refers to the index of the jth channel in C_i, i.e., $f_j^{(i)} \in \mathcal{M}$. Finally, we assume a time-slotted system for CRN communications. Time synchronization for the purpose of maintaining a single time reference can be achieved using any of the methods in [30, 44, 87], or by periodic synchronization to PR signals.

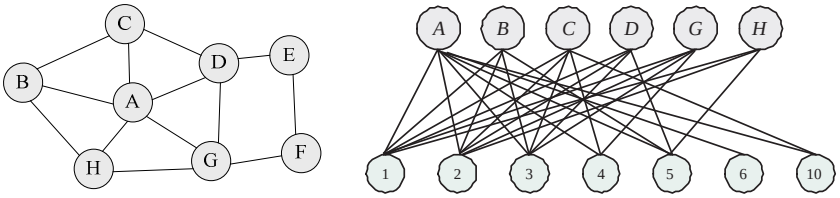
4.3 Cluster-based Channel Assignment

To account for space- and time-dependent spectrum availability, we propose a cluster-based CCA scheme. In this scheme, the CRN is partitioned into clusters. Nodes in a given cluster observe highly similar spectrum opportunities. The control channel within each cluster is selected from the set of common idle channels. Ensuring a large set of common idle channels in each cluster has several advantages. First, if the current control channel becomes occupied by a PR user, CRs can migrate to a new one. Second, grouping neighboring CRs with similar idle channels implicitly implements hard-decision cooperative sensing [33, 34, 58]. Third, multiple concurrent data transmissions can take place within each cluster. On the other hand, if spectrum opportunities are highly heterogeneous, requiring a large number of common idle channels per cluster may lead to small cluster sizes and a large number of clusters. In this case, cluster management and inter-cluster communications involve significant overhead.

To provide a graceful tradeoff between cluster size and the number of cluster-wide idle channels, we formulate the clustering problem as a *bipartite graph problem*. Specifically, our clustering algorithms, called Spectrum-Opportunity Clustering (SOC) and Constrained-SOC (C-SOC), utilize two instances of a biclique construction problem: the *maximum edge biclique graph problem* [24, 61], and the *maximum one-sided edge biclique graph problem* [23]. We first present the mapping from the clustering problem to a biclique construction. Then, we show how bicliques can be utilized for distributed clustering.

4.3.1 Mapping to Biclique Graphs

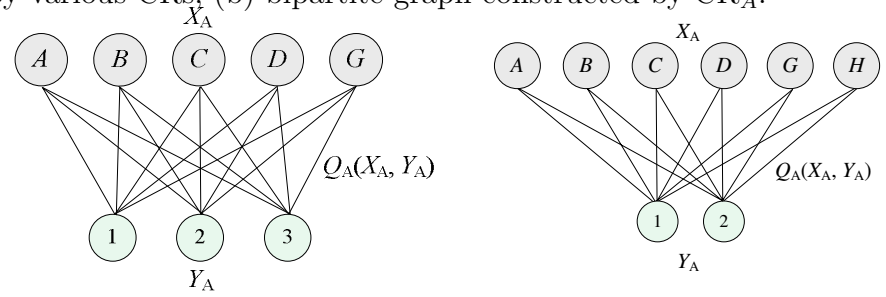
Figure 4.2(a) shows the connectivity graph of an example CRN. Following neighborhood discovery and the exchange of idle channels, each CR_i becomes aware of its one-hop neighbors CR_j and the channel list $C_j, \forall\ \mathrm{CR}_j \in \mathrm{NB}_i$. This information can be represented as a bipartite graph. A graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is called bipartite if the set of vertices $\mathcal{V}$ can be partitioned into two disjoint sets $\mathcal{A}$ and $\mathcal{B}$ with $\mathcal{A} \cup \mathcal{B} = \mathcal{V}$,



(a) (b)

$C_A = \{1, 2, 3, 4, 5, 6, 10\}$, $C_B = \{1, 2, 3, 5, 7\}$, $C_C = \{1, 2, 3, 4, 10\}$, $C_D = \{1, 2, 3, 5, 7\}$, $C_E = \{2, 3, 5, 7\}$, $C_F = \{2, 4, 5, 6, 7, 10\}$, $C_G = \{1, 2, 3, 4, 8\}$, $C_H = \{1, 2, 5, 8\}$.

Figure 4.2: (a) Connectivity graph of an 8-node CRN, and the lists of idle channels sensed by various CRs, (b) bipartite graph constructed by CR_A.



(a) (b)

$C_A = \{1, 2, 3, 4, 5, 6, 10\}$, $C_B = \{1, 2, 3, 5, 7\}$, $C_C = \{1, 2, 3, 4, 10\}$, $C_D = \{1, 2, 3, 5, 7\}$, $C_E = \{2, 3, 5, 7\}$, $C_F = \{2, 4, 5, 6, 7, 10\}$, $C_G = \{1, 2, 3, 4, 8\}$, $C_H = \{1, 2, 5, 8\}$.

Figure 4.3: Two possible bicliques for the bipartite graph $\mathcal{G}_A$: (a) Maximum-edge biclique constructed according to Algorithm 1, (b) maximum one-sided edge biclique constructed according to Algorithm 2.

such that every edge in $\mathcal{E}$ connects a vertex in $\mathcal{A}$ to a vertex in $\mathcal{B}$. For CR_i, the set $\mathcal{A}$ corresponds to its neighborhood set NB_i plus CR_i itself, while $\mathcal{B}$ corresponds to the set of idle channels C_i. An edge (x, y) exists between vertices $x \in \mathcal{A}_i$ and $y \in \mathcal{B}_i$ if $y \in C_x$, i.e., channel y is in the channel list of CR_x. In Figure 4.2(b), we show the bipartite graph $\mathcal{G}_A(\mathcal{A}_A \cup \mathcal{B}_A, \mathcal{E}_A)$ constructed by CR_A. By construction, CR_A is connected to all vertices in $\mathcal{B}_A$.

A bipartite graph $Q(\mathcal{V} = X \cup Y, \mathcal{E})$ is called a *biclique* if for each $x \in X$ and $y \in Y$ there exists an edge between x and y, i.e., $\mathcal{E} = \{(x, y) \mid \forall\, x \in X \text{ and } \forall\, y \in Y\}$. The edge set $\mathcal{E}$ is completely determined by X and Y, and hence, is omitted

from the biclique notation. For CR_i, a biclique graph $Q_i(X_i, Y_i)$ can be extracted from its bipartite graph $\mathcal{G}_i$. This biclique represents a cluster of nodes X_i that have channels $Y_i \subseteq C_i$ in common. In Figures 4.3(a), and 4.3(b) we show two possible bicliques for the bipartite graph $\mathcal{G}_A$. The first biclique (Figure 4.3(a)) represents the cluster $X_A = \{\mathrm{CR}_A, \mathrm{CR}_B, \mathrm{CR}_C, \mathrm{CR}_D, \mathrm{CR}_G\}$ with common channels $Y_A = \{1, 2, 3\}$. The second biclique (Figure 4.3(b)) represents the cluster $X_A = \{\mathrm{CR}_A, \mathrm{CR}_B, \mathrm{CR}_C, \mathrm{CR}_D, \mathrm{CR}_G, \mathrm{CR}_H\}$ with common channels $Y_A = \{1, 2\}$. The algorithms for obtaining both bicliques will be given shortly.

To organize the CRN into clusters, we are interested in forming bicliques that satisfy certain performance criteria. We choose to: (a) maximize the set of edges of the biclique graph, or (b) maximize the cluster size under a constraint on the number of common idle channels in the cluster. We now describe both criteria in detail.

4.3.2 Maximum Edge Biclique Graphs

The first clustering criterion is to maximize the number of edges in the biclique graph. This corresponds to maximizing the product of the cluster size and the number of common channels. We show that this criterion gracefully adapts to spatial heterogeneity in spectrum availability. It also provides a tradeoff between cluster size and the number of idle channels in a cluster.

To illustrate, consider the biclique $Q_i(X_i, Y_i)$ associated with CR_i. Suppose that there is another biclique $Q_i^*(X_i^*, Y_i^*)$, where $|X_i^*| = |X_i| + \Delta|X_i|$ and $|Y_i^*| = |Y_i| + \Delta|Y_i|$. Note that an increase in the number of common channels by $\Delta|Y_i| > 0$ will result in $\Delta|X_i|$ change in the cluster size, where $\Delta|X_i| \leq 0$. According to the maximization criterion, Q_i^* should be selected over Q_i if

$$|X_i||Y_i| < (|X_i| + \Delta|X_i|)(|Y_i| + \Delta|Y_i|). \tag{4.1}$$

The above inequality translates into:

$$-\left(\frac{\Delta|X_i|}{|X_i|} + \frac{\Delta|Y_i|}{|Y_i|}\right) \;<\; \frac{\Delta|X_i|\Delta|Y_i|}{|X_i||Y_i|} \tag{4.2}$$

under the constraints

$$|Y_i| + \Delta|Y_i| \leq |C_i| \quad \text{and} \quad |X_i| + \Delta|X_i| \leq |N_i|.$$

Inequality (4.2) states that the fractional change in the number of edges has to be larger than the fractional change in the number of vertices of the biclique. The effect of the maximum-edge biclique construction on the clustering process can be explained as follows. If the CRs in a given neighborhood have similar channels lists, our clustering rule will be fairly inclusive, resulting in large clusters. On the other hand, if the channel lists of neighboring CRs vary significantly, the clustering rule will reduce the cluster size in favor of the common channels within each cluster. Note that in general, the maximum edge biclique may contain any subset of vertices of the original bipartite graph. Our construction guarantees that: (a) any channel common to all neighbors of CR_i will be part of the maximum-edge biclique Q_i^*, and (b) CR_i will also be part of Q_i^*. This is shown in the following lemma.

Lemma 1. *Let a vertex $x \in \mathcal{A}$ of a bipartite graph $\mathcal{G}(\mathcal{A} \cup \mathcal{B}, \mathcal{E})$ be connected to all vertices in the set $\mathcal{B}$. Then, x belongs to the maximum-edge biclique $Q^*(X^*, Y^*)$.*

Proof. We prove Lemma 1 by contradiction. Let $x \in \mathcal{A}$ be a vertex of a bipartite graph $\mathcal{G}(\mathcal{A} \cup \mathcal{B}, \mathcal{E})$ such that there exists an edge $(x, y),\ \forall y \in \mathcal{B}$. Let $Q^*(X^*, Y^*)$ be the maximum edge biclique, and assume that $x \notin X^*$. By adding x to the graph Q^*, we obtain graph $Q'(X^* \bigcup x, Y^*)$, which is still a biclique since x is connected to every vertex in $\mathcal{B}$, and hence, every vertex in Y^*. The number of edges of the biclique Q' is $(|X^*| + 1) \times |Y^*| > |X^*| \times |Y^*|$. This contradicts our initial assumption that Q^* is a maximum-edge biclique. The same result can be shown for any vertex $y \in \mathcal{B}$ that is connected to all vertices in $\mathcal{A}$. □

Finding the maximum-edge biclique of a bipartite graph is an NP-complete problem [61]. For small bipartite graphs, an exhaustive search is possible. However, the search space grows exponentially with the cardinality of the vertex set. Accordingly, we develop a greedy heuristic (6) that produces a biclique with a large number of edges. In each iteration, Algorithm 6 examines one CR node. The vector S_i holds

Algorithm 6 Greedy Heuristic for Computing the Maximum Edge Biclique Graph

INPUT $\mathcal{G}_i(\mathcal{A}_i \cup \mathcal{B}_i, \mathcal{E}_i)$ // bipartite graph of CR_i
$Y_i \leftarrow \mathcal{B}_i$
for $j = 1$ to $|\mathcal{A}_i|$ **do**
  Find $\text{CR}_k \in \mathcal{A}_i$ that maximizes $|Y_i \bigcap C_k|$ over all nodes in $\mathcal{A}_i$
  if $Y_i \bigcap C_k = \emptyset$ **then**
    break
  else
    $S_i[j] = k$
    $\mathcal{A}_i \leftarrow \mathcal{A}_i - \text{CR}_k$, $X_i \leftarrow X_i \bigcup \text{CR}_k$, $Y_i \leftarrow Y_i \bigcap C_k$
    $P_i[j] = |X_i| \times |Y_i|$
  end if
end for
Find $j^* = \arg\max_j P_i[j]$
$Q^*(X_i^*, Y_i^*)$; $X_i^* = \{\text{CR}_{S_i[1]}, \ldots, \text{CR}_{S_i[j^*]}\}$; $Y_i^* = \bigcap_{k=1}^{k=j^*} C_{S_i[k]}$

the indices of CRs that have already been examined, whereas Y_i holds the list of common channels for the CRs in S_i. Initially, S_i is empty and $Y_i = \mathcal{B}_i$. In each iteration, we find a node CR_k whose channel list C_k has the highest overlap with Y_i. We then remove CR_k from $\mathcal{A}_i$, add k to S_i, and repeat the process until either $\mathcal{A}_i$ is empty or until the intersection of the common channel list with the remaining CRs is null. Note that in each iteration, $Q(X_i, Y_i)$ is a biclique. We record the number of edges of each constructed biclique in the vector P_i, and then find the biclique with the maximum number of edges. Algorithm 6 guarantees that a CR_x with $C_i \subseteq C_x$ will be included in the biclique Q_i^*. This is formalized in the following lemma.

Lemma 2. *Any $x \in A_i$ with $C_i \subseteq C_x$ will be included in the biclique $Q_i^*(X_i^*, Y_i^*)$ computed by Algorithm 6.*

Proof. Let $x \in \mathcal{A}_i$ be a vertex of a bipartite graph $\mathcal{G}_i(\mathcal{A}_i \cup \mathcal{B}_i, \mathcal{E}_i)$. Suppose that there exists an edge (x, y) $\forall y \in \mathcal{B}_i$. Assume that the maximum edge biclique $Q_i^*(X_i^*, Y_i^*)$ is computed during the jth iteration of Algorithm 1. Then any CR added to X_i in

the previous iterations will be part of Q_i^*. Hence, it is sufficient to show that x will be added to X_i^* before or during the jth iteration. If $Y_i^* = C_i$ then $x \in X_i^*$, since the addition of x increases the number of edges of Q_i^* by $|C_i|$. If $Y_i^* \subset C_i$, there exists some $x' \in X_i^*$ such that $C_{x'} \bigcap C_i \subset C_i$. Since on initialization $Y_i = C_i$ and $C_x \bigcap C_i = C_i$ according to line 4 of Algorithm 6, x will be added to X_i^* before x'. Hence, Q_i^* must contain x. □

We illustrate the steps of Algorithm 6 when executed at node A of Figure 4.2(a). The bipartite graph $\mathcal{G}_A$ for CR_A is shown in Figure 4.2(b). In the first iteration, node A is selected, as it has the highest overlap with $Y_A = C_A$. The number of edges of the biclique is $P_A[1] = 7$. In the second iteration, CR_C is selected, resulting in cluster $X_A = \{\mathrm{CR}_A, \mathrm{CR}_C\}$, $Y_A = \{1, 2, 3, 4, 10\}$, and $P_A[2] = 10$. Subsequent iterations result in the addition of $\mathrm{CR}_D, \mathrm{CR}_B, \mathrm{CR}_G$, and CR_H, in this order, and corresponding numbers of edges $P_A[3] = 9$, $P_A[4] = 12$, $P_A[5] = 15$, and $P_A[6] = 12$. The biclique Q_A^* with the maximum number of edges is eventually obtained. The final outcome is the cluster $\{\mathrm{CR}_A, \mathrm{CR}_B, \mathrm{CR}_C, \mathrm{CR}_D, \mathrm{CR}_G\}$ with common channels $\{1, 2, 3\}$, depicted in Figure 4.3(a).

4.3.3 Maximum One-Sided Edge Biclique Graphs

When maximizing the number of edges in the biclique, no requirement is imposed on the cluster size $|X_i|$ or the set of common channels $|Y_i|$. If there are large differences between the channel lists of neighboring CRs, this approach may result in clusters of very small sizes. To avoid this outcome, we examine a constrained version of the maximum-edge biclique problem, which aims at maximizing the cluster size while satisfying a lower bound on the number of common channels. Such a formulation is related to the maximum one-sided edge biclique problem [23], which can be stated as follows. Given a bipartite graph $\mathcal{G}(\mathcal{A} \cup \mathcal{B}, \mathcal{E})$ and a positive integer k, we wish to find a maximum-edge biclique with at least k nodes on one side of the bipartition. In our problem, this corresponds to imposing a lower bound on $|Y_i|$ and maximizing $|X_i|$.

Algorithm 7 Greedy Heuristic for Computing the Maximum One-sided Edge Biclique Graph

INPUT $\mathcal{G}_i(\mathcal{A}_i \cup \mathcal{B}_i, \mathcal{E}_i)$; t_0
$Y_i = \emptyset, \quad X_i = \mathcal{A}_i, \quad k = 1$
while $|Y_i| < \gamma_0$ and $|X_i| > 0$ **do**
  Find $y_k^* = \arg\max_{y \in \mathcal{B}_i \backslash Y_i} deg(y)$
  if y_k^* connects to no CR **then**
    break
  else
    $X_i \leftarrow X_i \bigcap S_i$; $S_i = \{\text{CR}_j \in \mathcal{A}_i \mid y_k^* \in C_j\}$
    $Y_i \leftarrow Y_i \bigcup y_k^*$
  end if
  $k = k + 1$
end while
$Q^*(X_i^*, Y_i^*)$

The maximum one-sided edge biclique problem is known to be NP-complete [23]. We provide a greedy algorithm (Algorithm 7) that yields clusters with $|Y_i| \geq \gamma_0$ idle channels in common, where γ_0 is a desired threshold. Algorithm 7 examines one channel in each iteration. The set X_i is initialized to all one-hop neighbors of CR_i. The set Y_i is initially empty. At the kth iteration, CR_i finds channel $y_k^* \in B_i \backslash Y_i$ that is common to the largest number of one-hop neighbors, i.e., the channel with the highest connectivity degree $deg(y)$ in the bipartite graph $\mathcal{G}_i(\mathcal{A}_i \cup \mathcal{B}_i, \mathcal{E}_i)$. Any neighbor that has not sensed y_k^* as idle is removed from X_i. Then, y_k^* is added to Y_i. If several channels have the same degree, the decision of selecting y_k^* is deferred to the next iteration. All values of y_k^* are stored and for each one we find channel $y_{k+1}^* \in B_i \backslash Y_i + y_k^*$ with the highest degree. Assuming y_{k+1}^* is unique, then y_k^* and y_{k+1}^* are added to Y_i; else we proceed to the next iteration[1]. Note that at each step,

[1]To simplify the exposition, the details of breaking the tie are not shown in the pseudo-code of Algorithm 7.

the graph $Q(X_i, Y_i)$ is a biclique, because y_k^* is connected to all CRs in X_i. The number of required iterations in Algorithm 7 is equal to or less than γ_0.

We illustrate the application of Algorithm 7 to the CRN of Figure 4.2(a). Let $\gamma_0 = 2$. For node A, we initially have Y_A=$\emptyset$ and X_A=$\{\text{CR}_A, \text{CR}_B, \text{CR}_C, \text{CR}_D, \text{CR}_H, \text{CR}_G\}$. Channels 1 and 2 have the highest degree of six, so both of them are included in Y_A. The cluster membership X_A remains intact. In the next round, we start with Y_A=$\{1\}$ and X_A=$\{\text{CR}_A, \text{CR}_B, \text{CR}_C, \text{CR}_D, \text{CR}_H, \text{CR}_G\}$ and add channel 2 to Y_A. X_A remains the same. For Y_A=$\{2\}$ and X_A=$\{\text{CR}_A, \text{CR}_B, \text{CR}_C, \text{CR}_D, \text{CR}_H, \text{CR}_G\}$, channel 1 is selected. So the two (X_A, Y_A) pairs form the same biclique, given by Y_A=$\{1, 2\}$ and X_A=$\{\text{CR}_A, \text{CR}_B, \text{CR}_C, \text{CR}_D, \text{CR}_H, \text{CR}_G\}$. Since $|Y_A|$ satisfies $\gamma_0 \geq 2$, the algorithm terminates and returns $Q_A(X_A, Y_A)$. This biclique is depicted in Figure 4.3(b).

4.4 Spectrum-Opportunity Clustering

In this section, we develop the *spectrum-opportunity clustering* (SOC) algorithm which utilizes the clustering criteria based on the biclique mapping, as presented in Section 4.3. The SOC algorithm follows four steps:

Step 1: *Biclique computation*–In step 1, each CR_i is aware of its one-hop neighbors along with the channel availability at each $\text{CR}_j \in \text{NB}_i$. Using this information, CR_i constructs a bipartite graph and computes the "best" biclique $Q_i^1(X_i^1, Y_i^1)$. Here the superscript is used to denote the obtained biclique after a given iteration. The "best" biclique is computed using Algorithm 6 or Algorithm 7, as described in Section 4.3. We refer to the clustering method that uses Algorithm 6 as SOC, and to the clustering method that uses Algorithm 7 as Constrained SOC (C-SOC). Once the optimal bicliques are computed, CR_i broadcasts its biclique info $Q_i^1(X_i^1, Y_i^1)$ to its neighbors.

Step 2: *Updating cluster memberships*–In step 2, each CR_i checks if there is a biclique Q_j^1 with $\text{CR}_i \in X_j^1$ that provides *better* clustering than Q_i^1. That is, it checks if $Q_j^1 > Q_i^1$ with $\text{CR}_i \in X_j^1$. The inequality operator for two bicliques is

defined as follows.

Definition 9. *For two bicliques $Q_i(X_i, Y_i)$ and $Q_j(X_j, Y_j)$ constructed using Algorithm 1, we say $Q_i < Q_j$ if:*

(a) $|X_i| \times |Y_i| < |X_j| \times |Y_j|$, *or*

(b) $|X_i| \times |Y_i| = |X_j| \times |Y_j|$ *and* $|X_i| < |X_j|$, *or*

(c) $|X_i| = |X_j|, \quad |Y_i| = |Y_j|$, *and* $i < j$.

In Definition 9, we first compare the number of edges in the two bicliques. If two bicliques have the same number of edges, we then compare their cluster sizes. If the cluster sizes are also equal, we break the tie by selecting the biclique of the CR with the highest id. Definition 9 imposes a total ordering between two bicliques (i.e., two bicliques can never be equal). Now for the C-SOC case, the inequality operator is defined as follows.

Definition 10. *For two bicliques $Q_i(X_i, Y_i)$ and $Q_j(X_j, Y_j)$ constructed using Algorithm 7, we say $Q_i < Q_j$ if:*

(a) $|X_i| < |X_j|$, *or*

(b) $|X_i| = |X_j|$ *and* $|Y_i| < |Y_j|$, *or*

(c) $|X_i| = |X_j|, \quad |Y_i| = |Y_j|$, *and* $i < j$.

In Definition 10, given that both Q_i and Q_j satisfy the constraint on the number of idle channels, we select the biclique that leads to a larger cluster size. If cluster sizes are equal, we compare the number of idle channels per cluster. If those are equal as well, we break the tie by selecting the biclique of the CR with the higher id. CR_i selects biclique $Q_j^1(X_j^1, Y_j^1)$ with $\text{CR}_i \in X_j^1$, that is best according to the relation operator given in Definitions 9 or 10, and updates its maximum edge biclique to $Q_i^2 = Q_j^1$. After computing Q_i^2, CR_i informs its neighbors of the updated cluster membership X_i^2 and the common channel list Y_i^2.

We illustrate the execution of step 2 for the CRN in Figure 4.2(a). CR_A receives the following updates from its neighbors: (a) Q_B^1 with $X_B = \{\mathrm{CR}_A, \mathrm{CR}_B, \mathrm{CR}_H\}$ and $Y_B = \{1, 2, 5\}$, (b) Q_C^1 with $X_C = \{\mathrm{CR}_A, \mathrm{CR}_B, \mathrm{CR}_C, \mathrm{CR}_D\}$ and $Y_C = \{1, 2, 3\}$, (c) Q_D^1 with $X_D = \{\mathrm{CR}_A, \mathrm{CR}_C, \mathrm{CR}_D, \mathrm{CR}_E, \mathrm{CR}_G\}$ and $Y_D = \{2, 3\}$, (d) Q_G^1 with $X_G = \{\mathrm{CR}_A, \mathrm{CR}_D, \mathrm{CR}_G, \mathrm{CR}_H\}$ and $Y_G = \{1, 2\}$, and (e) Q_H^1 with $X_H = \{\mathrm{CR}_A, \mathrm{CR}_B, \mathrm{CR}_G, \mathrm{CR}_H\}$ and $Y_H = \{1, 2\}$. Ordering the bicliques according to Definition 9 yields $Q_G^1 < Q_H^1 < Q_B^1 < Q_D^1 < Q_C^1 < Q_A^1$. Then A sets $Q_A^2 = Q_A^1$ since Q_A^1 has the maximum number of edges. Similarly, $Q_B^2 = Q_A^1$, $Q_C^2 = Q_A^1$, $Q_D^2 = Q_A^1$, $Q_G^2 = Q_A^1$, and $Q_H^2 = Q_A^1$.

Step 3: *Finalizing cluster membership*–In step 3, each CR_i examines the cluster membership X_i^2. For each $\mathrm{CR}_j \in X_i^2$, if $\mathrm{CR}_i \notin X_j^2$ then CR_i removes CR_j from its biclique Q_i^2. At the completion of this step, the final bicliques Q_i^3 are obtained for all nodes that were not removed from the biclique of their choice in step 2. If a CR_i has adopted the clustering of CR_j during step 2, it may be the case that X_j^2 contains CRs not within the range of CR_i. For those CRs, CR_i will not have biclique information. To provide this information, every CR_j must replay all received biclique information if its biclique is selected by any of its neighbors in step 2.

For illustration, consider the CRN in Figure 4.2(a). CR_A checks if it is included in the bicliques of all CRs in X_A^2. Given that all neighboring CRs have adopted Q_A^1, CR_A concludes that $X_A^3 = X_A^2$ and $Y_A^3 = Y_A^2$. Similarly, B goes through its list $X_B^2 = \{\mathrm{CR}_A, \mathrm{CR}_B, \mathrm{CR}_C, \mathrm{CR}_D, \mathrm{CR}_G\}$. Because CR_D and CR_G are not neighbors of CR_B, the only way that CR_B can know about Q_D^2 and Q_G^2 is if CR_A relays their updates. According to our algorithm, CR_A relays Q_D^2 and Q_G^2, because both CR_D and CR_G have adopted Q_A^1. CR_B can now see that it is included in the bicliques of CR_D and CR_G, so it sets $X_B^3 = \{\mathrm{CR}_A, \mathrm{CR}_B, \mathrm{CR}_C, \mathrm{CR}_D, \mathrm{CR}_G\}$. Note that CR_H is excluded from the biclique updates of $\mathrm{CR}_A, \mathrm{CR}_B$, and CR_G, and therefore continues to step 4.

Step 4: *Unclustered CRs*–CRs that did not join any clusters because they were removed from the biclique they chose in step 3 repeat steps 1-3, but exclude already

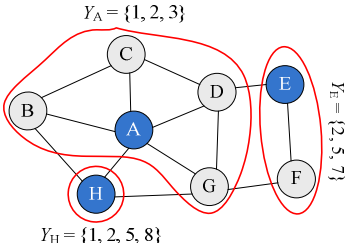


Figure 4.4: Final clustering based on SOC. CR_A, CR_E, and CR_H are the CHs.

clustered neighbors. For example, CR_H in Figure 4.2(a) was excluded by the biclique of CR_A and hence, has to join another cluster. CR_H deletes CR_A, CR_B and CR_G from its neighbor list, and exchanges information with the remaining neighbors to construct a cluster. If there are no remaining one-hop neighbors, as in the case of CR_H, then a single CR cluster is formed. The final clustering and the list of common channels are shown in Figure 4.4.

4.4.1 Correctness of the SOC Algorithm

We now prove that the SOC algorithm leads to consistent cluster memberships, i.e. all CRs distributively reach the same clustering outcome. The correctness proof follows the logic of the clique clustering method in [79]. Several modifications are made to the use of bicliques with cluster memberships that possibly do not form cliques. To show the correctness of SOC, we prove that at the end of step 3, $Q_i^3 = Q_j^3$ for any CR_j that belongs to cluster X_i^3. Because step 4 is a repetition of steps 1-3, it follows that SOC converges to the same cluster memberships. We first prove a series of lemmas leading to our main proof.

Lemma 3. *If* $CR_i \in X_j^2$ *and* $CR_j \in X_i^2$, *then* $Q_i^2 = Q_j^2$.

Proof. After step 1, CR_i and CR_j will have received the updates of their neighbors. Suppose that CR_i selects $Q_i^2 = Q_k^1$, where CR_k is a neighbor of CR_i, or is CR_i itself. Given that $\text{CR}_j \in X_i^2$, then $\text{CR}_j \in X_k^1$, and hence, CR_j is a neighbor of CR_k. Following a similar argument, we can show that for the decision $Q_j^2 = Q_m^1$

to be made, the selected Q_j^2 must be constructed by a node $\mathrm{CR}_m \in \mathrm{NB}_i$, given that $\mathrm{CR}_i \in X_j^2$. Because CR_k and CR_m are neighbors of both CR_i and CR_j, CR_i and CR_j must have received both Q_k^1 and Q_m^1 in step 1, before updating their own bicliques. Due to the imposed total ordering, CR_i concludes that $Q_m^1 < Q_k^1$, and CR_j concludes that $Q_k^1 < Q_m^1$. This is true only if $k = m$. □

According to Lemma 3, two CRs that include each other in their respective bicliques after step 2 must have agreed on the same bicliques. We utilize this result in Lemma 4.

Lemma 4. *Suppose that for three nodes CR_i, CR_j, and CR_k, we have $CR_k \in X_i^2$ and $CR_k \in X_j^2$ with $Q_i^2 = Q_j^2$. Then if $CR_i \notin X_k^2$, it must also hold that $CR_j \notin X_k^2$.*

Proof. Lemma 4 can be proved by contradiction. Assume that $\mathrm{CR}_j \in X_k^2$. Because $\mathrm{CR}_j \in X_k^2$ and $\mathrm{CR}_k \in X_j^2$, then $Q_j^2 = Q_k^2$ by Lemma 3. However, by assumption we also have $Q_i^2 = Q_j^2$, and hence $Q_i^2 = Q_k^2$. Since $\mathrm{CR}_k \in X_i^2$ and $Q_i^2 = Q_k^2$, this also means that $\mathrm{CR}_i \in X_k^2$, which leads to a contradiction. Hence, $\mathrm{CR}_j \notin X_k^2$. □

Based on Lemmas 3 and 4, we now show that SOC guarantees that CRs will have consistent cluster membership information. It also follows that CRs will agree on the set of common channels.

Theorem 2. *For any $CR_j \in X_i^3$, $Q_i^3 = Q_j^3$.*

Proof. Q_i^3 is a pruned version of Q_i^2, i.e., $X_i^3 \subseteq X_i^2$. Therefore, any $\mathrm{CR}_j \in X_i^3$ must also be a member of X_i^2. Also for any $\mathrm{CR}_j \in X_i^3$, we have $\mathrm{CR}_i \in X_j^2$, since otherwise, CR_j would have been removed from X_i^3. Using Lemma 3, it follows that $Q_i^2 = Q_j^2$. Now consider any $\mathrm{CR}_k \in X_i^2$ that is removed from X_i^2 in step 3, i.e., $\mathrm{CR}_k \notin X_i^3$. This happens only if $\mathrm{CR}_i \notin X_k^2$, which also means (by Lemma 2) that $\mathrm{CR}_j \notin X_k^2$, and CR_k will also be removed from X_j^2 in step 3. Hence, every CR that is removed from X_i^2 will also be removed from X_j^2, making $X_i^3 = X_j^3$. For two bicliques with the same membership, it follows that $Y_i^3 = Y_j^3$, and hence $Q_i^3 = Q_j^3$. □

Based on Theorem 2, at the end of step 3, CRs that have not been excluded from their cluster choice in step 2 agree on the same clusters. For any CRs that are removed (CR_H in our example), steps 1-3 are repeated with the exclusion of any already clustered members. Hence, the new clusters formed at step 4 lead to consistent cluster formation.

4.4.2 Clusterhead Election

SOC is a cluster-first algorithm, so clusterheads (CHs) are elected after clusters are formed. CHs are used to facilitate operations such as cooperative sensing, routing, and topology management. A typical requirement for a CH is that it must be connected to all members of its cluster. In SOC, though CRs of a cluster are not guaranteed to form a clique (for example in Figure 4.3(a), CR_B, CR_D are not within each other's communication range even though they belong to the same cluster), in the following lemma we prove that at least one CR in the cluster is guaranteed to be within range of every member of its cluster. This CR can be identified in the last step of cluster formation.

Lemma 5. *In every cluster produced by SOC, at least one CR is one-hop away from all other CRs of that cluster.*

Proof. Consider a cluster that is represented by the biclique $Q_i^3(X_i^3, Y_i^3)$. According to Theorem 2, all $\text{CR}_j \in X_i^3$ converge to the same cluster membership in step 3. For any CR_i and $\text{CR}_j \in X_i^3$, it holds that $\text{CR}_i \in X_j^2$ and $\text{CR}_j \in X_i^2$. Otherwise, CR_i would have removed CR_j from X_i^2 in step 2, and similarly CR_j would have removed CR_i from X_j^2. According to Lemma 3, if $\text{CR}_i \in X_j^2$ and $\text{CR}_j \in X_i^2$, it holds that $Q_i^2 = Q_j^2$. This means that all members of a cluster formed after step 3 must have computed the same biclique in step 2. However, the biclique Q_i^2 of any CR in step 2 is the best biclique Q_j^1 with $\text{CR}_j \in \text{NB}_i$ or $j = i$. Hence, the only way that all CRs of a cluster would choose Q_j^1 as the best biclique in step 2 is if CR_j is a neighbor to all. Therefore, at least one CR is one hop away from all CRs of the cluster. □

For the CRN in Figure 4.4, CR_A, CR_E, and CR_H can be selected as CHs.

4.5 Dynamic Control Channel Assignment

Once clusters are formed, control channels must be selected from the common idle channel list within each cluster. This assignment can be facilitated by CHs. From an architectural standpoint, the assignment of different control channels to various clusters poses two major challenges.

Inter-cluster coordination problem–Consider the clustering in Figure 4.4. Suppose that CR_A, CR_E, and CR_H serve as CHs. For the three formed clusters, supposed that channels 1, 7 and 8 are selected for control respectively. Assume now that CR_G wants to send a control message to CR_F. Since channel 7 is not in the idle list of CR_G, the two CRs cannot exchange control messages despite the fact that channels $\{2, 4\}$ are common to both of them.

Control channel migration problem–For the CRN in Figure 4.4, suppose that a PR starts transmitting over channel 1, and only CR_B senses this PR activity (other CRs may be out of range of the transmitting PR or may not be sensing the channel). CR_B needs to notify the other CRs in its cluster that channel 1 is no longer idle. Since channel 1 is used for control, a notification sent on this channel will interfere with the PR transmission. Even if this interference is considered negligible due to its short duration, CR communication on channel 1 may not be possible due to the PR activity. To migrate the control channel, the CH node CR_A has to correctly receive the notification from CR_B and determine a new control channel for the nodes in its cluster.

4.5.1 Periodic Control-Channel Rotation

To allow for inter-cluster communication and to coordinate control-channel migration, we propose the following periodic channel-rotation mechanism. Rather than selecting one channel for control until PR activity appears on it, the control channel is rotated among the common idle channels within each cluster. Let $W_i = \{f_1^{(i)}, f_2^{(i)}, \ldots, f_{|W_i|}^{(i)}\}$ denote the set of common channels in cluster i. For time slots $t = 1, 2, \ldots,$ CRs within cluster i use channel $f_j^{(i)}$ where $j =$

$[(t-1) \pmod{|W_i|}] + 1$. The channel hopping mechanism is similar to the hopping used in the neighbor discovery mechanism. However, CRs hop only through the list of common channels within their cluster and channel schedules between clusters may be different. To illustrate, consider the CRN of Figure 4.4. For the cluster $\{\mathrm{CR}_A, \mathrm{CR}_B, \mathrm{CR}_C, \mathrm{CR}_D, \mathrm{CR}_G\}$, the set of common channels is $W_A = \{1, 2, 3\}$. The (slot, control channel) pairs for the first few slots are (1,1), (2,2), (3,3), (4,1)... Similarly, for cluster $\{\mathrm{CR}_E, \mathrm{CR}_F\}$, the (slot, control channel) pairs are (1,2), (2,5), (3,7), (4,2)...

When a CR senses PR activity on the current control channel, it waits until the control channel is migrated to an idle one before notifying other CRs within the same cluster. For example, in Figure 4.4, suppose that CR_B senses PR activity on channel 1. When the control channel hops to channel 2, CR_B notifies its CH (CR_A) of its new idle channel list. Then, CR_A updates the list of common channels to $W'_A = \{2, 3\}$ and broadcasts W'_A, using either channel 2 or 3. The control channel now rotates only between channels 2, and 3.

The rotation of the control channel also addresses the problem of inter-cluster coordination. Two neighboring CRs that belong to two different clusters can communicate with each other as long as the two clusters have at least one idle channel in common. For example, if CR_G is aware of the common channel list $W_E = \{2, 5, 7\}$ of the cluster $\{E, F\}$, it can use time slots t, where $(t-1) \equiv 1 \pmod{3} + 1$, to communicate control information on channel 2. To enable inter-cluster coordination, CRs use the broadcast of Q_j^3 from their neighbors to obtain the common channel list of adjacent clusters and derive their channel schedule. The above rotation mechanism implements an always-on virtual channel for control, located at different frequency bands in various time slots and clusters. The location of the control channel is known to all CRs within each cluster.

4.5.2 Reclustering

Although SOC converges after only a few messages are exchanged, it is desirable to limit frequent recomputation of clusters in order to reduce the communication

overhead for forming new clusters, the traffic relay, and temporary disconnections. In SOC, the availability of multiple idle channels reduces the need for reclustering. The set of common idle channels is updated in accordance with PR activity. However, it may happen that a cluster is left without any common channel for some time, due to low idle channel availability. If this time is small, cluster members may temporarily switch to the coordination protocol until sufficient channels are freed again. A reclustering operation can be triggered periodically to account for the long-term dynamics of PR activity and changes in the CRN topology. In Section 4.7.3, we show that in SOC, only a small fraction of clusters are left without any common idle channel.

4.6 Coordination Without a Control Channel

During the execution of the SOC algorithm, neighboring CRs need to exchange their lists of idle channels. This exchange has to occur in the absence of a common control channel, because such a channel is not yet established. In this section, we propose a coordination protocol for CRNs that facilitates the exchange of broadcast information. Our mechanism relies on a combination of well established principles of multiple access such as time division and random access. Note that several coordination mechanisms that do not require the existence of a control channel are known for fixed spectrum networks (e.g., [6, 76]), but their adaptation to CRNs is not straightforward. The work most relevant to ours is the quorum channel hopping (QCH) system proposed in [11]. We compare the performance of our protocol with the scheme in [11], in Section 4.7.6.

4.6.1 Protocol Overview

Consider an arbitrary node CR_i. The steps of our coordination protocol are as follows:

1. CR_i determines C_i using spectrum sensing.

2. CR_i broadcasts its list C_i on channel $f_j^{(i)} \in C_i$ during slots $t = 1, 2, ...,$ if the following relation is satisfied: $f_j^{(i)} = [(t-1) \pmod M] + 1$. Broadcasting is done according to a random access protocol. Any CR_ℓ that hears CR_i's transmission places CR_i in its neighbor list, denoted as NB_ℓ.

3. CR_i exchanges clustering information (explain in Section 4.4) with every neighbor $\text{CR}_\ell \in \text{NB}_i$ using the channel schedule derived from C_ℓ (e.g. on channel $f_j^{(i)}$ at time slot $kM + f_j^{(i)}, k = 0, 1, ...,$ if they both see channel $f_j^{(i)}$ as available), until a common control channel is set up.

In Step 2, a universal time schedule for channel access is followed, regardless of nodes' individual views of channel availability. Each time slot t is mapped to a channel $j \in \mathcal{M}$ by a modulo-M operation. For example, for the CRN in Figure 4.1, the (slot, channel) pairs during which the broadcast in Step 2 is allowed to take place are (1,1), (2,2) (3,3), (4,4), (5,1),... CR_A can communicate with CR_C and CR_D on channel 3 at time slots $t = 3, 7, 11,...$ Using this universal schedule, CRs can discover their neighbors and exchange channel information.

Our neighbor discovery protocol requires all CRs to be time-synchronized. For the purpose of neighbor discovery, a time slot corresponds to the time that CRs operate on one idle channel. The length of a time slot can be made appropriately large, to allow for the discovery of all CRs that are tuned to the same channel. Access to each frequency band can occur in a random fashion following a CSMA model [1]. Though random access protocols have relatively low throughput, they are preferred here because they do not require any node coordination.

Each time slot is divided into mini-slots. Each mini-slot is long enough to allow CR_i to broadcast its list of idle channels. CRs tuned to the same channel broadcast their channel lists at each mini-slot with probability P_{access}. If a CR chooses to stay silent in a given mini-slot (with probability $1 - P_{access}$), it will listen to other transmissions and record their announced channel lists. If more than one CR chooses the same mini-slot for transmission, a collision occurs. Note that in typical wireless communications, broadcast messages are not acknowledged in order to avoid the

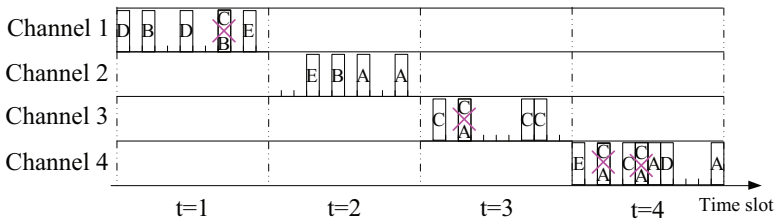


Figure 4.5: A realization of the coordination protocol for the CRN of Figure 4.1.

ACK implosion problem. Because of the absence of feedback regarding the reception of a broadcast message, contending nodes continue to access mini-slots with probability P_{access}, regardless of the success of their transmission.

One realization of the above coordination process for the CRN of Figure 4.1 is shown in Figure 4.5. There are four available channels. Each time slot is divided into 12 mini-slots. For time slot 1, we have $[(1-1) \pmod 4] + 1 = 1$, so nodes whose channel lists include channel 1 (CR_D, CR_B and CR_E) tune to this channel. These nodes contend at various mini-slots in slot 1. Even though a collision occurs during the 9th mini-slot, CR_D, CR_B and CR_E are still able to successfully broadcast their available channels in mini-slots 1,3,6,11 of slot 1. During time slot 2, CR_A, CR_B and CR_E tune to channel two, and announce their available channels without any collision. All of them identify each other as neighbors. This channel access mechanism is maintained until a control channel is established.

4.7 Performance Evaluation

In this section, we demonstrate the agility of our clustering algorithms in adapting to PR dynamics. We first investigate the performance of C-SOC for different threshold values. Then, we demonstrate the advantages of SOC and C-SOC over clustering methods that do not take into account PR activity. Moreover, we evaluate the rate of reclustering due to PR activity for various clustering methods. We then evaluate the performance of the periodic control-channel rotation mechanism, and study the effectiveness of Algorithms 6 and 7 in finding bicliques that are close to the optimal ones. Finally, we evaluate the performance of the coordination protocol, and compare it with the QCH system proposed in [11].

4.7.1 Evaluation Setup

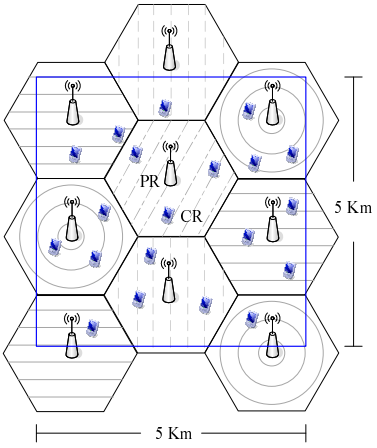


Figure 4.6: Evaluation setup consisting of a cellular PRN and a CRN. Ten channels are assigned per cell. Adjacent cells do not share any channels.

In our evaluation, we consider a CRN that co-exists with a cellular PRN. The parameters for each of the two networks as well as the evaluation metrics are described below.

Cellular primary network setup–The cellular network consists of nine cell towers covering an area of 5Km×5Km, as shown in Figure 4.6. 40 frequency channels are assigned to the PRN, according to the four-color theorem [69]. Accordingly, each cell is assigned 10 channels, with adjacent cells operating over non-overlapping channels. This is illustrated in Figure 4.6 by the different shading on the various cells. The communication range for each cell tower is set to 1.25 Km. For each cell, calls arrive at each channel according to a Poisson process of rate λ calls/min. We assume an exponentially distributed call holding time with parameter μ minutes.

CRN Setup–CRs are randomly deployed in the area covered by the cellular network. They are assumed to be fixed. The CR communication range is set to 500 m. Each CR_i senses the set of idle channels C_i based on the cell it is located in. A CR is not allowed to access channels occupied by the cell it resides in or by adjacent cells. An imperfect sensing process is assumed, whereby the status of a channel at each CR is misdetected with probability p_f. The lists of idle channels are updated at each CR every time a new event (call arrival or call termination) occurs in a cell.

Clustering Schemes–We compare SOC and C-SOC against three clustering schemes: (a) the distributed clustering algorithm (DCA) [10], (b) the lowest-id clustering algorithm (LCA) [9], and (c) the distributed coordination scheme proposed in [94]. We will refer to the latter as DCRN. In DCA, a node is elected as a CH if it has the highest weight among its neighbors. Each CR associates itself with a neighboring CH that has the highest weight. We set the weight of a CR to its degree on the connectivity graph. In LCA, a node becomes a CH if it has the lowest id within its neighborhood. Finally, in DCRN, CRs select the channel common to the largest number of neighboring CRs for control. CRs may belong to multiple clusters at the same time, if they utilize more than one control channel to connect to their neighbors.

Evaluation Metrics:

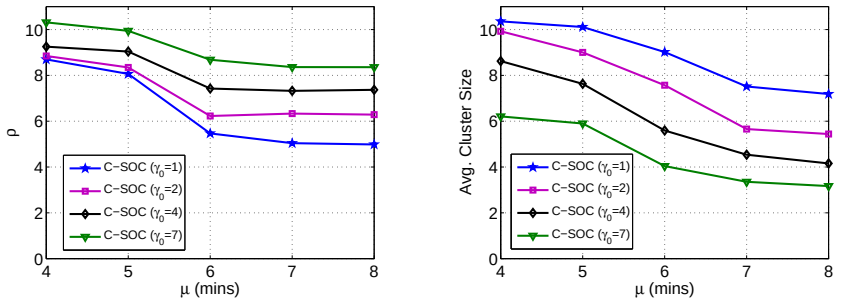
- *Average number of common idle channels per cluster*–This metric, denoted by ρ, captures channel availability in a cluster. A larger value of ρ enables control-channel migration (in case of PR activity) with less likelihood of reclustering. It also implies that higher bandwidth is available for intra-cluster communications.

- *Coefficient of variation (CV) for the number of common idle channels*–The CV is defined as the ratio of the standard deviation over the mean value. This metric captures the uniformity on the availability of common idle channels per cluster. A low CV implies more uniformity among clusters.

- *Percentage of clusters with no common idle channels*–Poor clustering decisions can lead to clusters with no common idle channels. This metric captures the percentage of clusters whose members do not share any channels.

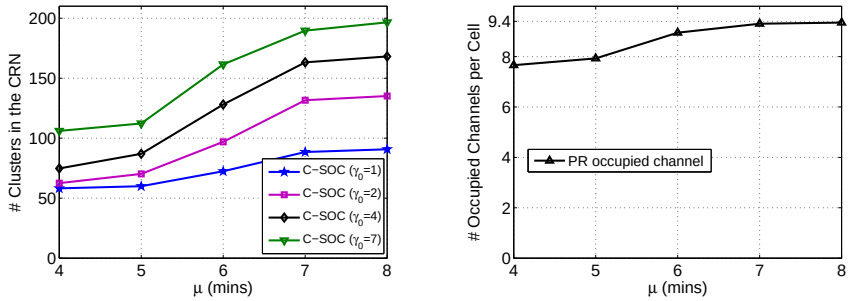
- *Number of clusters in the network*–The partitioning of the CRN into a large number of clusters increases the overhead for inter-cluster coordination.

- *Average cluster size*–This metric represents the average number of CRs that belong to a cluster.



(a) (b)

Figure 4.7: Performance of C-SOC as a function of the call duration μ for different values of γ_0: (a) average number of common channels per cluster (ρ), (b) average cluster size



(a) (b)

Figure 4.8: Performance of C-SOC as a function of the call duration μ for different values of γ_0: (a)average number of clusters in the CRN, and (b) number of occupied channels by PR per cell.

- *CV of the average cluster size*–This metric captures the uniformity of the constructed clusters in terms of number of nodes per cluster.

4.7.2 Evaluation of the C-SOC Algorithm

In this set of experiments, 600 CRs are deployed. We fix λ at 1.5 calls/min, and vary μ from $\mu = 4$ mins to $\mu = 8$ mins. We first investigate the effect of the threshold value γ_0 on the number of common idle channels per cluster. As shown in Figure 4.7(a), ρ is larger than γ_0 for every value of μ. This effect can be explained by two factors. For small μ's, nodes within the same neighborhood share many more channels than γ_0. The threshold γ_0 does not have a significant effect on the

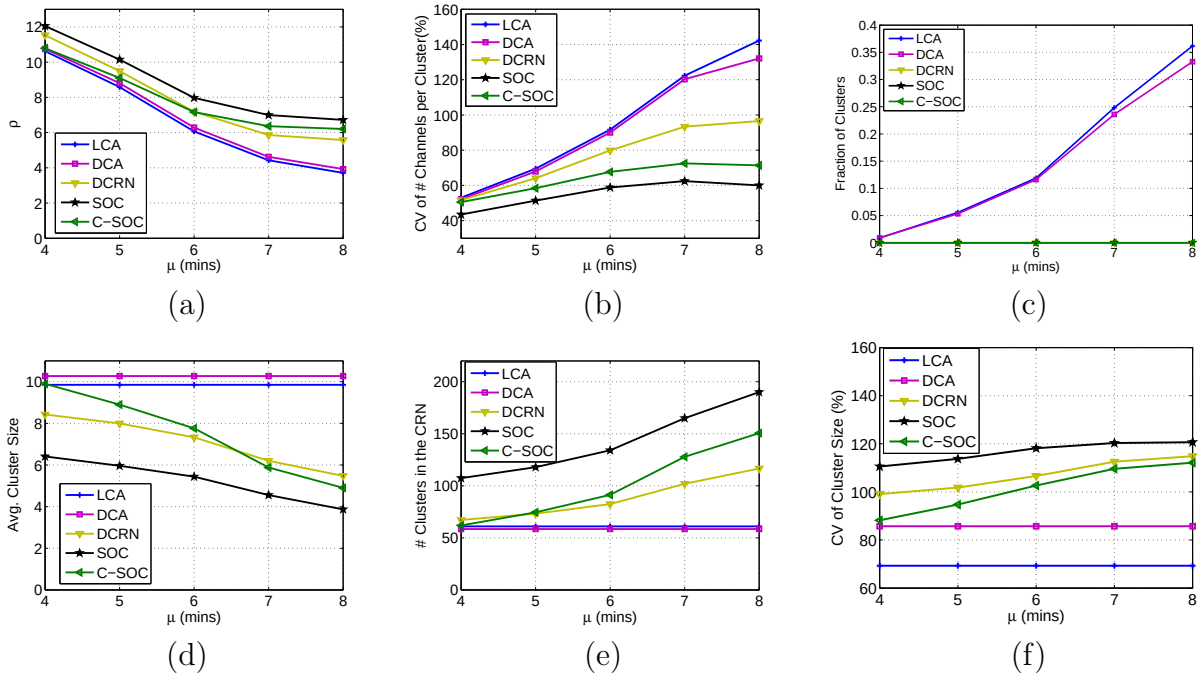


Figure 4.9: Performance of various clustering schemes vs. call duration μ: (a) average number of common idle channels per cluster, (b) CV of the number of common channels, (c) fraction of clusters with no common idle channels, (d) average cluster size, (e) average number of clusters in the CRN, (f) CV of the cluster size.

clustering process. Thus, the neighborhood size is the limiting factor in cluster size. Indeed, in Figure 4.7(b), we observe very small differences in the cluster sizes for different γ_0. The impact of γ_0 becomes apparent when γ_0 is large (e.g., $\gamma_0 = 7$) or when μ is large. As μ increases, fewer channels become available for CR user, and hence, the common idle channel availability within each cluster drops. Nonetheless, ρ remains above γ_0 for all values of μ.

The threshold requirement in C-SOC has adverse impact on the cluster size and the number of clusters. To maintain ρ above γ_0, C-SOC creates smaller size clusters, leading to a partitioning of the CRN into a large number of clusters. This is depicted in Figure 4.8(a). In fact, for $\gamma_0 = 7$, the number of clusters is almost twice as large when $\mu = 8$ mins compared to $\mu = 4$ mins. This increase can be explained in conjunction with Figure 4.8(b), which shows the PR channel occupancy as a function of μ. We observe that for large μ, there are not enough idle channels to

satisfy a high γ_0. This leads to the creation of many single-node clusters.

4.7.3 Comparison of SOC/C-SOC with Other Schemes

Variation in PR activity

In this set of experiments, we vary the PR activity by varying the average call duration μ. In Figure 4.9(a), we depict ρ as a function of μ. The threshold for C-SOC is set to $\gamma_0 = 2$. We observe that SOC and C-SOC maintain a larger value of ρ for all μ's. This advantage is demonstrated in the CV for the number of idle channels, and the fraction of clusters with no common channels. Figure 4.9(b) shows this CV as a function of μ. We observe a much smaller variation in channel availability for SOC and C-SOC compared with the other schemes. This behavior is essential to guarantee sufficient idle channels for migration when a PR appears on the current control channel. Hence, frequent reclustering is avoided.

When PR activity is high, clustering methods that do not take into account channel availability create many clusters with no common idle channels. This is illustrated in Figure 4.9(c), which shows the fraction of clusters with no common idle channels as a function of μ. We observe that for $\mu = 8$ mins, 35% of the clusters created using LCA and DCA do not share any common idle channels. SOC, C-SOC, and DCRN mitigate this problem, due to their spectrum-aware nature.

To increase ρ, SOC and C-SOC adjust the cluster size. This is verified in Figures 4.9(d) and 4.9(e), which depict the average cluster size and the number of clusters in the CRN as a function of μ, respectively. We observe that the average cluster size decreases with μ to compensate for the reduction in idle channel availability. In turn, this leads to the creation of more clusters, as shown in Figure 4.9(e). Figure 4.9(f) shows the CV for the cluster size under various schemes. A higher variation is observed for SOC, C-SOC and DCRN. This is due to the spatial adaptation of the cluster size to PR activity. One critical observation is that C-SOC behaves like DCA for low μ, yielding large cluster sizes and low CV values. when PR activity is low.

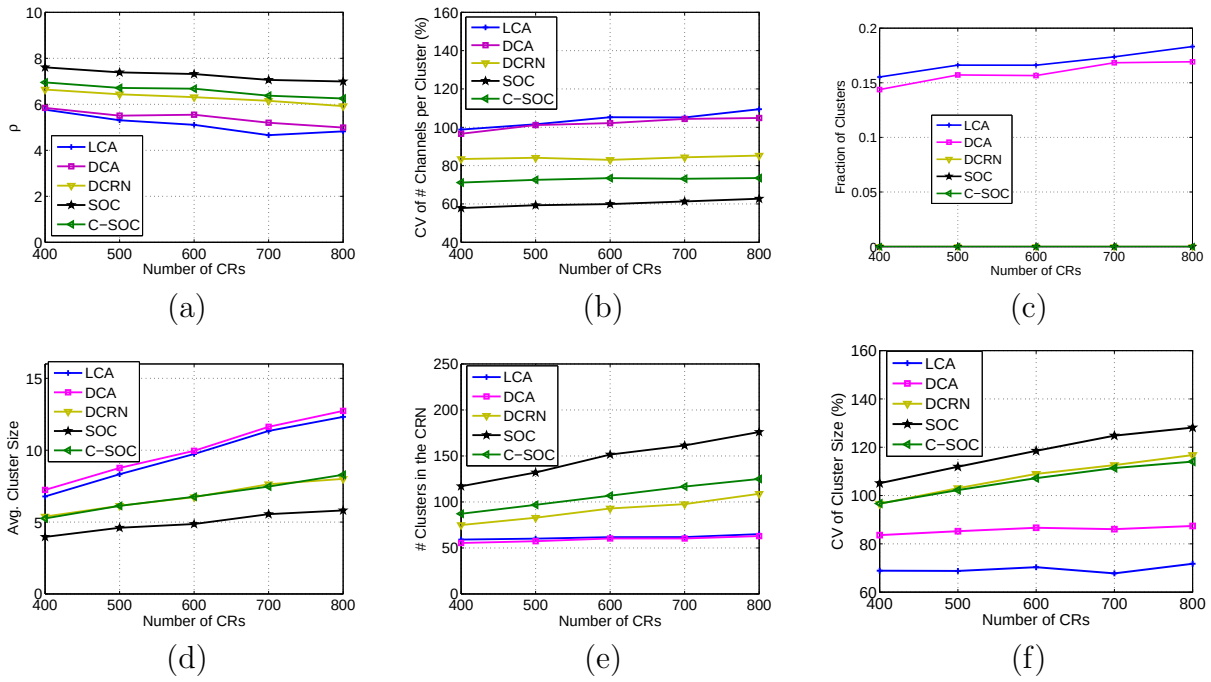


Figure 4.10: Performance of various clustering schemes vs. node density: (a) average number of common idle channels per cluster, (b) CV of the number of common channels, (c) fraction of clusters with no common idle channels, (d) average cluster size, (e) average number of clusters in the CRN, (d) CV of the cluster size.

Variation in Node Density

In this set of experiments, we vary the node density by varying the number of deployed CRs. The call arrival rate is set to $\lambda = 1.5$ calls/min per cell, with the mean call duration fixed to 6 mins. Higher node density leads to more flexibility in clustering, which can be potentially exploited to maintain a large number of common idle channels per cluster. In Figure 4.10(a), we depict ρ as a function of the number of CRs. Observe that for spectrum-aware solutions, ρ is almost insensitive to changes in node density, while decreasing for LCA and DCA. As shown in Figure 4.10(b), the CV for the common channels under SOC and C-SOC remains constant with variations in node density. Moreover, Figure 4.10(c) shows that for the same PR activity, the fraction of clusters with no common idle channels increases with the number of deployed CRs for LCA and DCA, up to about 18%. SOC, C-SOC, and

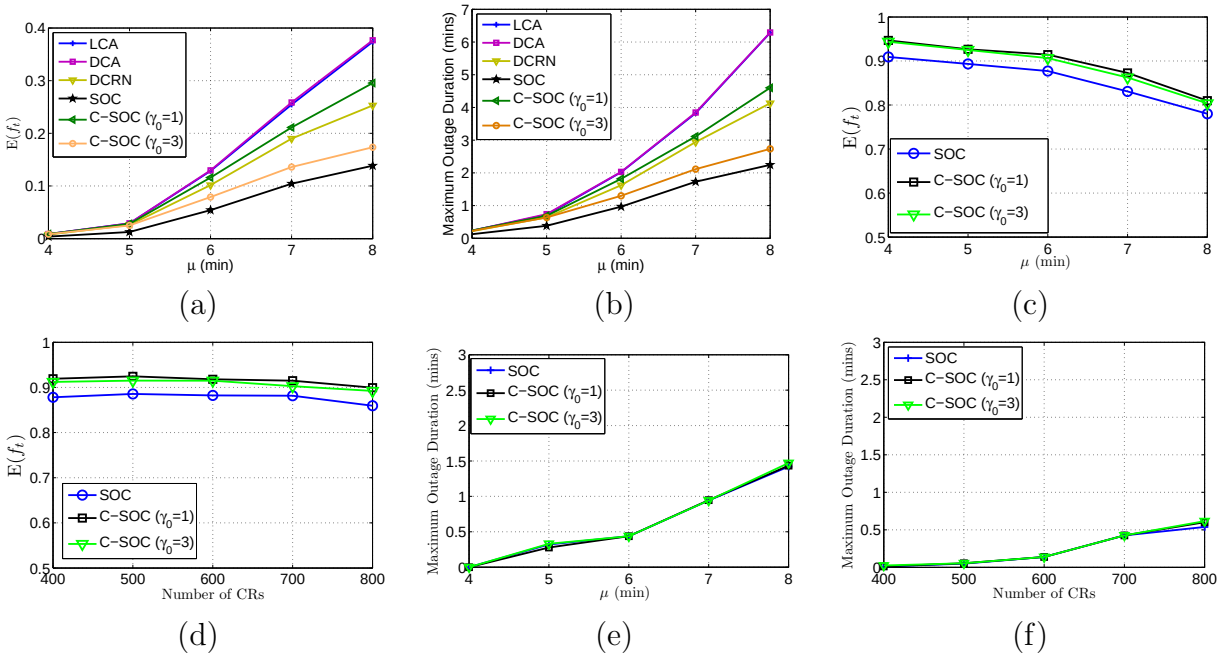


Figure 4.11: (a) Fraction of time that at least one cluster exists without a common idle channel as a function of μ, (b) maximum duration of the control channel outage as a function of μ, (c) $\mathrm{E}(f_t)$ as a function of μ, (d) $\mathrm{E}(f_t)$ as a function of the number of CRs, (e) maximum outage duration for inter-cluster communication as a function of μ, and (f) maximum outage duration for inter-cluster communication as a function of the number of CRs.

DCRN do not produce any cluster with no common idle channels.

In Figures 4.10(d),(e),(f), we respectively show the average cluster size, the number of clusters in the CRN, and the CV for the cluster size, all as functions of the number of deployed CRs. Under SOC and C-SOC, the number of clusters increases almost linearly with the number of deployed CRs, in order to maintain a stable value of ρ. Moreover, our algorithms exhibit a larger variability in cluster size, as they adapt to variations in spectrum opportunities.

Frequency of Reclustering

In this set of experiments, we investigate the rate of reclustering due to PR activity. We first partition the CRN into clusters based on a snapshot of the channel

availability. We then fix λ to 1.5 calls/min and vary the value of μ.

In Figure 4.11(a), we show the average fraction of time $E(f_t)$ where *at least* one cluster with no common idle channels exists. We observe that as the PR activity increases, several clusters created based on a snapshot of PR activity stay without a control channel. SOC and C-SOC provide the best performance out of all clustering algorithms. Even when $\mu = 8$ mins, every cluster created using SOC has at least one common idle channel 86% of the time, in contrast to a 63% for LCA and DCA, and 75% for DCRN. C-SOC with $\gamma_0 = 1$ behaves almost as DCRN since only one idle channel is required per cluster. C-SOC yields a significant improvement for $\gamma_0 = 3$, for which 83% of the time clusters have at least one common idle channel.

Reclustering is greatly impacted by the outage duration of the control channel, defined as the period of time that cluster stays without a common idle channel. Figure 4.11(b) shows the maximum outage duration as a function of μ. We observe that SOC and C-SOC have much shorter outage durations compared with other schemes, thus avoiding reclustering. Note that for $\mu = 8$ almost all channels within each cell are occupied by the PR, as shown in Figure 4.7(d). Hence, the outage is mainly caused by the lack of available channels, rather than poor clustering decisions.

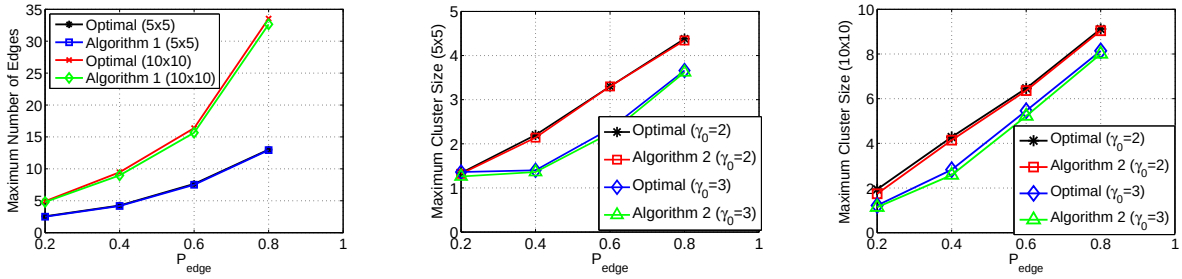
4.7.4 Periodic Control-Channel Rotation

In this set of experiments, we evaluate the periodic control-channel rotation mechanism which is employed for inter-cluster communications. In Figure 4.11(c), we show the average fraction of time $E(f_t)$ that a cluster is reachable by CRs in adjacent clusters, as a function of μ when $\lambda = 2$ calls/min. We observe that when the PR activity is low (small values of μ), our rotation mechanism can maintain inter-cluster communication more than 90% of the time. This is because under low PR activity, the majority of the channels used for channel rotation within a cluster are also idle in adjacent clusters. On the other hand, under high PR activity (large values of μ), the heterogeneity between the sets of idle channels of neighboring CRs increases. To adjust to this change, SOC creates clusters of smaller size in order to maintain a larger set of common idle channels within each cluster. In this case,

the overlap between the set of common idle channels of a cluster and the set of idle channels of CRs in adjacent clusters tends to be smaller. Nevertheless, over 78% of the time, inter-cluster communication is still feasible. Note that when μ is large, there are periods of time where inter-cluster communication is not possible simply because all available channels are occupied by PRs. For such periods of high activity, inter-cluster communication is limited by the opportunistic nature of the CRNs and is not related to the channel rotation mechanism.

From Figure 4.11(c), we also observe that C-SOC with $\gamma_0 = 1$ yields a higher value of $\mathrm{E}(f_t)$ compared to the case of $\gamma_0 = 3$ and to SOC. This difference can be explained as follows. For small values of γ_0, the network partitioning is primarily decided by its physical topology. To maximize the cluster size, C-SOC produces clusters with a small number of common idle channels. These channels are, therefore, likely to be seen idle by a large number of CRs, including CRs in adjacent clusters. Restricting channel rotation to those widely available channels increases the fraction of time that inter-cluster communication is possible. On the other hand, in SOC, it is more likely that during channel rotation, a channel is common among the cluster members but is not available to CRs in adjacent clusters (this is the reason why these members were excluded from that cluster in the first place). In Figure 4.11(d), we show $\mathrm{E}(f_t)$ as a function of the number of CRs in the network We observe that $\mathrm{E}(f_t)$ almost remains constant with the increase of the number of CRs. This is due to the fact that the spatial variation in PR activity is not affected by the number of deployed CRs.

Figures 4.11(e) and 4.11(f) show the maximum outage duration in inter-cluster communications, measured as the maximum time (in minutes) that *any* two adjacent clusters are unable to communicate during the course of the simulation. Outages in inter-cluster communications can occur when: (a) a cluster is left without any common idle channels due to PR activity, or (b) the set of common idle channels used for channel rotation does not overlap with the sets of idle channels for CRs in an adjacent cluster. In our simulations, we only measured outages due to scenario (b), since outages due to (a) are already evaluated in Section 4.7.3. From



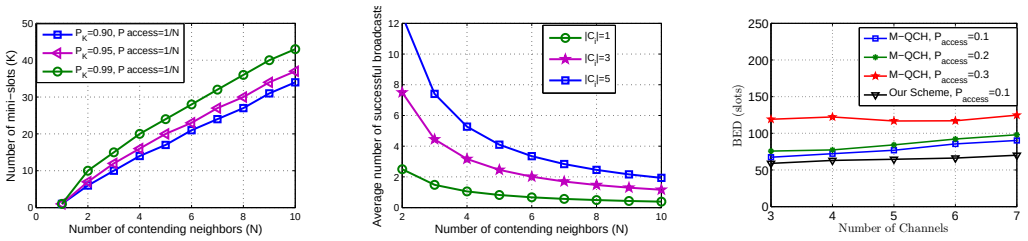
(a) (b) (c)

Figure 4.12: Comparison of Algorithms 6 and 7 with the optimal solution as a function of the probability of edge existence P_{edge}: (a) Algorithm 6 for 5×5 and 10×10 bipartite graphs, (b) Algorithm 7 for 5×5 bipartite graphs and for $\gamma_0 = 2, 3$, (c) Algorithm 7 for 10×10 bipartite graphs and for $\gamma_0 = 2, 3$.

Figure 4.11(e), we observe that the maximum outage time is limited to small values (<1.5 minutes) even when $\mu = 8$ min. Moreover, Figure 4.11(f) illustrates a slightly increasing maximum outage time when the CR density is increased. This is because the probability of finding a CR that does not share common channels with an adjacent cluster increases with the increase of the number of border nodes.

4.7.5 Performance of Algorithms 6 and 7

In this section, we evaluate the performance of our greedy heuristics. Because the problems of finding the maximum edge biclique and the maximum one-sided node cardinality are NP-complete, we obtain optimal solutions via exhaustive search for small bipartite graphs. We randomly generate bipartite graphs of sizes 5×5 and 10×10. In each bipartite graph, an edge between a pair of vertices exists independently with probability P_{edge}, which is varied from 0.2 to 0.8. Though our experiments are limited in the size of the bicliques, they are are nonetheless useful for typical neighborhood sizes and number of channels. In Figure 4.12(a), we show the number of edges in bicliques obtained by Algorithm 6 when performing an exhaustive search, averaged over 50 bipartite graphs, as a function of P_{edge}. We observe that for the considered bipartite graphs, our greedy heuristic is near-optimal. In Figure 4.12(b),(c) we show the cluster size obtained by Algorithm 7 and through exhaustive



(a) (b) (c)

Figure 4.13: (a) Number of mini-slots versus N for a given P_K with $P_{access} = 1/N$, (b) expected number of successful broadcasts after M time slots versus N ($K = 10$), (c) the BED as a function of the number of available channels under a dynamic spectrum scenario with $\lambda = 2$ calls/min and $\mu = 0.5$ mins.

search, averaged over 50 bipartite graphs, as a function of P_{edge}, and for threshold values $\gamma_0 = 2, 3$. Once again, our greedy heuristic is near-optimal.

4.7.6 Evaluation of The Coordination Protocol

Analytical evaluation

We first compute the number of mini-slots needed, so that on average, every contending CR is able to perform one successful broadcast per time slot. Let N be the number of contending neighbors and let K be the number of mini-slots per time slot. Without loss of generality, we take $N < K$. Let P_{mini} denote the probability of a successful transmission in a given mini-slot. Because each contending node independently accesses a mini-slot with probability P_{access}, $P_{mini} = \binom{N}{1} P_{access} (1 - P_{access})^{N-1}$. Simple calculations show that P_{mini} is maximized when $P_{access} = 1/N$, i.e., when on average, exactly one CR attempts a transmission in a mini-slot. The probability of at least N successes in K mini-slots is:

$$P_K = \sum_{i=N}^{K} \binom{K}{i} P_{mini}^i (1 - P_{mini})^{K-i}. \tag{4.3}$$

One possible way to select K is to impose a lower bound on P_K. In Figure 4.13(a), we plot K as a function of N for different values of P_K, with P_{access} set to $1/N$. We observe that the minimum number of mini-slots that is necessary to

guarantee a certain P_K increases approximately linearly with N. Approximately 15 mini-slots are enough when $N = 5$ with $P_K = 0.9$, while 35 mini-slots are needed when $N = 10$.

The neighbor discovery phase ends after M time slots, at which point all possible channels will have been scanned. The number of successful broadcasts for a single node CR_i in M time slots is binomially distributed with mean of $K|C_i|P_{access}\left(1 - P_{access}\right)^{N-1}$.

Figure 4.13(b) shows this mean for different $|C_i|$, when $K = 10$. When CR_i has more than three available channels, it can make at least one successful broadcast even if there are nine other contending CRs. When on average only one channel is available, CR_i can still broadcast successfully if less than four CRs are contending. In this case, a large value of K is required.

Although in general channel availability is not expected to change during the neighbor discovery phase, a CR may still detect new PR activity on channel j during time slot t. In this case, the CR vacates this channel, updates its channel list, and uses other idle channels to continue exchanging channel information. After one-hop neighbors are discovered, CRs use the time schedules of their neighbors to coordinate the clustering and CCA process.

Comparison to the QCH system

Similar to our scheme, the QCH system proposed in [11] can be used for the coordination of CRs in the absence of a control channel. To compare the performance between the two schemes, we have measured the number of slots needed until every node within the same collision domain communicates one message to all its neighbors (our control channel assignment scheme requires that all neighbors exchange their set of idle channels during the coordination phase). We refer to this delay as the *broadcast exchange delay* (BED).

To provide a fair comparison with the QCH scheme, we have considered the quorum-based design that is most suitable for broadcast communications. According to [11], the channel access delay is minimized when the channel hopping system is

constructed based on a majority cyclic quorum. Therefore, the authors suggest using the M-QCH system for the implementation of broadcast control channels. M-QCH has a minimum frame length equal to three and is $k = 3$ and is constructed over $U = Z_3$. We have selected the quorum system for constructing the hopping sequences as $S = \{\{0, 1\}, \{0, 2\}, \{1, 2\}\}$ for our simulations. In this setup, each node is randomly assigned a quorum from S in order to construct its hopping sequence.

To compare the performance of the two schemes under a dynamic spectrum scenario, we considered a PR network in which calls arrive at each channel according to a Poisson process of rate $\lambda = 2$ calls/min. We assumed an exponentially distributed call holding time with parameter $\mu = 0.5$ mins. Figure 4.13(c) shows the BED as a function of the number of available channels. We observe that our scheme is more efficient than the M-QCH scheme (M-QCH needs around 30% more time-slots to finish broadcast when the number of available channels is large). This difference in performance is attributed to the fact that in our scheme, CRs converge on the same channel in every time slot, thus facilitating the broadcast operation. In contrast, in M-QCH only a subset of nodes converge to the same channel at any time slot.

4.8 Conclusions

We addressed the problem of CCA in CRNs. We adopted a dynamic allocation policy in which the control channel is dynamically assigned according to PR activity. We mapped the clustering problem into instances of a bipartite graph problem, and showed that this mapping allows for a graceful tradeoff between the cluster size and the set of common channels in each cluster. In particular, we mapped the clustering process to the maximum edge biclique problem, and the maximum one-sided edge cardinality problem. Since both problems are known to be NP-complete, we proposed two greedy heuristics for finding bicliques that satisfy our requirements. We proposed two distributed clustering algorithms called SOC and C-SOC that takes into account the channel availability in deciding cluster memberships. We further proposed a control channel rotation mechanism that enables control channel

migration in case of PR activity, inter-cluster communication, and adaptation to the temporal variations of spectrum availability.

CHAPTER 5

Related Work

5.1 Jamming Attack in Wireless Networks

Jamming in wireless networks has been extensively studied. Most prior research assumes that the jammer is an external entity, oblivious to the protocol specifics and cryptographic secrets [73]. Recently, several works have considered the problem of jamming by an internal adversary, who exploits knowledge of network protocols and secrets to launch DoS attacks on layers above the physical layer [17, 55, 66, 67, 78, 80, 81]. In this section, we classify related work based on the adversarial model.

Jamming Under an Internal Threat Model– Chan et al. considered the problem of control-channel jamming in the context of GMS networks [17]. They proposed the replication of control information over multiple channels according to a binary encoding based key (BBK) assignment. Assuming an adversary who is capable of jamming only one channel per time slot, the authors derived necessary conditions to guarantee control channel access to all users within several slots. They also showed that the BBK assignment leads to the identification of a certain number of compromised nodes.

Tague et al. proposed a cryptographic key-based mechanism for hiding the control-channel slots [81]. Nodes can only discover a subset of these locations with some probability. Their method allows for graceful degradation in the control-channel secrecy as a function of the number of compromised nodes, as opposed to the threshold approach in [17]. Further, they proposed an algorithm called GUIDE for identifying compromised nodes based on the set of jammed control channels. They formulated the identification problem as a maximum likelihood estimation problem [81]. All methods in [17, 80, 81] consider a server-client model, where base stations are assumed to be secure.

Chiang et al. proposed an anti-jamming scheme for broadcast communications in DS- and FH-CDMA systems [21]. Their method organizes broadcast PN codes into a binary key tree. Each node on the tree corresponds to a unique PN code, known only to a subset of users. Every message is spread by multiple PN codes such that all users can decode using exactly one code. Identification of compromised nodes is achieved by relating the PN code adopted by the jammer to those known to each user. Desmedt et al. proposed an anti-jamming scheme that protects SS based broadcast communication [26]. With a known number of malicious insiders, they design the distribution of frequency allocation to prevent legitimate nodes from being jammed.

Several schemes eliminate the need for secret PN codes [8, 55, 66, 78]. Baird et al. proposed the BBC algorithm, which can recover jammed messages under some special conditions. can insert arbitrary messages into the broadcast channel but cannot erase any of the original messages. Pöpper et al. proposed a solution called Uncoordinated DSSS (UDSSS) [66]. In their scheme, broadcast transmissions are spread according to a PN code that is randomly selected from a public set of codes. At the receiving end, nodes have to record transmitted messages and attempt to decode them by exhaustively applying every PN code in the public codebook. Because the selected PN code is not known a priori to any receiver, the jammer has to guess the PN code, thus significantly complicating the jamming task. However, message transmissions have to be repeated several times to allow receivers to synchronize with the transmitter. Strasser et al. proposed an uncoordinated frequency hopping (UFH) scheme for establishing shared secret keys between devices that do not share any prior secrets, in the presence of a jammer [78]. In UHF, the transmitter and receiver hop between channels at random. After some number of hops, they are able to exchange a common pairwise key and independently derive a pairwise shared PN code. An improvement in communication latency and jamming resistance of the original UHF scheme was presented in [77], by combining coding techniques with hashing. Slater et al. improved the communication efficiency of UFH by using Merkle trees, distillation codes, and erasure coding [75].

Liu et al. proposed RD-DSSS, a randomized differential DSSS scheme that enables jamming-resistant broadcast using only publicly known PN codes [55]. In RD-DSSS, a "0" bit is encoded using two randomly selected PN codes with low correlation, while a "1" bit is encoded using two PN codes with high correlation. The selected PN codes are appended at the end of each message, thus slightly decreasing the communication efficiency compared with the original DSSS. Recovery of the PN codes that were selected by the sender is achieved only after the transmitted message is received.

Several methods attempt to identify the compromised nodes that leaked information to the jammer. Lazos et al. proposed the assignment of unique frequency hopping sequences to each receiver, overlapping in a fixed subset of hops [48]. Using the uniqueness of the assigned sequences, compromised nodes whose sequences are used for jamming are identified. Tague et al. proposed the GUIDE scheme for identifying compromised nodes based on the set of control channels that are jammed. They formulated the identification problem as a maximum likelihood estimation problem [81]. Chiang and Yih-Chun Hu, developed a code-tree based approach for identifying compromised PN codes [22].

Jamming Under an External Threat Model– Under an external threat model, jamming is often mitigated by employing SS techniques [73, 74]. In these techniques, the transmitted narrowband signal is spread over a larger bandwidth according to a secret PN code. Anti-jamming properties are achieved because more energy is required to cause interference in a larger bandwidth. The typical processing gain in SS communications is in the range of 20 to 30 dB [73, 74].

Xu et al. studied the problem of jamming in systems where spreading is not possible (or effective) [89, 91, 92]. They studied the problem of detecting physical-layer and MAC-layer DoS attacks based on jamming [92]. They proposed a slow frequency hopping method to avoid jamming, but assumed that hopping sequences remain secret. For mobile networks, they proposed the use of spatial retreats to avoid communication within the jammed area. Formal measures for detecting jamming attacks were introduced in [91]. Xu et al. also proposed the establishment of

a timing-based low bitrate covert channel to notify nodes outside the jamming area about the presence of a jammer [89]. This channel maps the inter-arrival times of corrupted packets into bits. Cagalj et al. proposed wormhole-based anti-jamming techniques for sensor networks [16]. Using a wormhole link, sensors within a jammed region establish communications outside this region, and notify them regarding ongoing jamming attacks.

Jamming Beyond the PHY Layer– The use of jamming as a vehicle for launching DoS attacks against higher-layer functionalities was studied in [14, 16, 45, 50, 51, 53, 56, 67]. Brown et al. demonstrated that a jammer can exploit implicit packet identifiers such as packet size, timing, and sequence number at the transport or network layer to classify transmitted packets and launch selective jamming attacks [14]. Proaño and Lazos showed the feasibility of selective jamming by performing real-time packet classification. Liu et al. proposed a layered architecture called SPREAD to mitigate the impact of smart jammers that target multiple layers of the network stack [53]. SPREAD randomizes protocols at each layer, thus increasing the adversary's uncertainty with respect to the protocol execution. Finally, Li. et al. provided a game theoretic approach to optimal jamming and anti-jamming strategies at the MAC layer [50].

5.2 Control Channel Assignment

Previously proposed CCA schemes for CRNs can be classified into: (a) static assignment of a dedicated frequency band common to all CRs, and (b) dynamic assignment based on criteria such as spatial correlation, spectrum usage, connectivity degree, etc. We describe both categories in detail.

Static Control Channel Assignment Schemes–Several researchers have proposed the exchange of control information on an always available static frequency band, known to all nodes (e.g., [13, 15, 36, 70]). Čabrić et al. proposed the CORVUS system, in which control traffic is transmitted using UWB technology [15]. Brown proposed the use of ISM bands for control in CRNs [13]. Han et al. proposed an

OFDM-based scheme to allow for long-range transmission of control messages with small bit error rates [36]. Several MAC protocol designs for CRNs assume the existence of a dedicated control channel, without specifying its allocation (e.g. [39, 70]).

Dynamic Control Channel Assignment Schemes–Zhao et al. proposed distributed coordination of CRs via a locally computed control channel that changes in response to PR activity [94]. The band available to the largest set of one-hop neighbors is selected for control in each neighborhood, implementing a partition of the CRN into clusters. This approach minimizes the number of distinct frequency bands needed for control, thus reducing the overhead of cluster management. However, this can lead to frequent reclustering due to variations in PR activity. Another cluster-based design was adopted in [20].

Chen et al. proposed a swarm-intelligence-based algorithm for adapting the control channel based on individual interference measurements [19]. Neighboring CRs engage in a negotiation process to decide on a control channel. This negotiation is carried out in licensed bands without consideration for PRs. Kondareddy and Agrawal proposed dynamic hopping of the control channel based on pseudo-random sequences [43]. Transmitter/receiver pairs randomly meet in different bands and decide on a common hopping sequence, called the *rendezvous channel*, until their data exchange is completed. One limitation of this design is that hopping coordination occurs over licensed channels without considering possible interference to PRs.

Bahl et al. proposed WhiteFi, a system that provides Wi-Fi like connectivity over the UHF white spaces [7]. WhiteFi incorporates dynamic channel assignment algorithms for detecting and managing spectrum opportunities. Similar to our scheme, WhiteFi handles control traffic in-band, using one main and one backup control channel. The locations of the main and backup channels vary according to the dynamics of the spectrum. The main differences between how WhiteFi handles the CCA problem and our work are that: (a) WhiteFi is designed for an access point-client architecture, where a large set of clients is connected to a single access point. We consider an ad hoc network model where the control channel has to be maintained over multiple hops; (b) WhiteFi maintains only one backup channel for

broadcasting control information. SOC organizes the CRN to clusters where several common idle channels are available for carrying control traffic and therefore, is more resilient to temporal variations of the spectrum.

CHAPTER 6

Conclusions and Future Work

6.1 Conclusions

In this dissertation, we have studied how to design jamming-resistent mechanisms under node compromise attack. In particular, we have focused on mechanisms for control-channel jamming attack, and control channel assignment problem in cognitive radio networks. Our main achievements and findings are summarized as follows.

First, we have studied the problem of jamming attacks in wireless network, and various attacks existing in each layer of a OSI model. For control-channel jamming attack, we proposed a randomized distributed scheme for maintaining and establishing the control channel using frequency hopping. Our method differs from classical frequency hopping in that the communicating nodes are not synchronized to the same hopping sequence. Instead, each node follows a unique hopping sequence. We further proposed a mechanism for adjusting hopping sequences to dynamic spectrum conditions without incurring any extra overhead. Our scheme can identify compromised nodes through their unique sequences and exclude them from the network. We evaluated the performance of our scheme both in static- and dynamic-spectrum networks, based on the metrics of evasion entropy, evasion delay, and evasion ratio. We further evaluated the Hamming distance between the jamming sequence and those assigned to compromised and uncompromised nodes. Our proposed scheme can be utilized as a temporary solution for re-establishing the control channel until the jammer and the compromised nodes are removed from the network.

Second, We have proposed TDBS, a scheme for jamming-resistant broadcast communications in the presence of inside jammers. In TDBS, broadcast is realized as a series of unicast transmissions distributed in frequency and time. Because

the adversary is limited in the number of channels he can jam, several unicast transmissions remain interference-free. We mapped the problem of constructing hopping sequences for TDBS to the problem of 1-factorization of complete graphs. We analytically evaluated the security properties of TDBS under an external and an internal threat model and showed that TDBS maintains broadcast communications even when multiple nodes are compromised. We verified our theoretical analysis using extensive simulations.

Third, we addressed the problem of CCA in CRNs. We adopted a dynamic allocation policy in which the control channel is dynamically assigned according to PR activity. We mapped the clustering problem into instances of a bipartite graph problem, and showed that this mapping allows for a graceful tradeoff between the cluster size and the set of common channels in each cluster. In particular, we mapped the clustering process to the maximum edge biclique problem, and the maximum one-sided edge cardinality problem. Since both problems are known to be NP-complete, we proposed two greedy heuristics for finding bicliques that satisfy our requirements. We proposed two distributed clustering algorithms called SOC and C-SOC that takes into account the channel availability in deciding cluster memberships. We further proposed a control channel rotation mechanism that enables control channel migration in case of PR activity, inter-cluster communication, and adaptation to the temporal variations of spectrum availability.

Lastly, we present related works to address jamming attacks in wireless network both under external treat model and internal threat model. Also works on CCA (control channel assignment) in static spectrum network and dynamic spectrum network are discussed respectively.

6.1.1 Future Work

REFERENCES

[1] N. Abramson. The aloha system: another alternative for computer communications. In *Proc. of AFIPS*, volume 70, pages 281–285, November 1970.

[2] D. Adamy. *EW 101: A first course in electronic warfare*. Artech House Publishers, 2001.

[3] I. Akyildiz, W. Lee, M. Vuran, and S. Mohanty. Next generation dynamic spectrum access cognitive radio wireless networks: A survey. *Computer Networks*, 50(13):2127–2159, 2006.

[4] M. Al-Shurman, S.-M. Yoo, and S. Park. Black hole attack in mobile ad hoc networks. In *Proc. of the 42nd annual Southeast regional conference*, pages 96 – 97, 2004.

[5] K. Appel and W. Haken. Every planar map is four colorable: part i. *Illinois journal of Mathematics*, 21(3):491–567, 1977.

[6] P. Bahl, R. Chandra, and J. Dunagan. SSCH: Slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad-hoc wireless networks. In *Proc. of (MobiCom)*, pages 216–230, 2004.

[7] P. Bahl, R. Chandra, T. Moscibroda, R. Murty, and M. Welsh. White space networking with Wi-Fi like connectivity. In *Proc. of the ACM SIGCOMM 2009 Conference*, pages 27–38, 2009.

[8] L. C. Baird, W. L. Bahn, M. D. Collins, M. C. Carlisle, and S. C. Butler. Keyless jam resistance. In *Proceedings of the 2007 IEEE Workshop on Information Assurance United States Military Academy*, 2007.

[9] D. Baker, A. Ephremides, and J. Flynn. The design and simulation of a mobile radio network with distributed control. *IEEE Journal on Selected Areas in Communications*, 2(1):226–237, 1984.

[10] S. Basagni. Distributed clustering for ad hoc networks. In *Proc. of I-SPAN*, pages 310–315, June 1999.

[11] K. Bian, J. Park, and R. Chen. A quorum-based framework for establishing control channels in dynamic spectrum access networks. In *Proc. of MobiCom*, pages 25–36, 2009.

[12] K. Bicakci and B. Tavli. Denial-of-service attacks and countermeasures in ieee 802.11 wireless networks. *Journal Computer Standards and Interfaces*, 31(5):931–941, 2009.

[13] T. Brown. An analysis of unlicensed device operation in licensed broadcast service bands. In *Proc. of DySPAN*, pages 11–29, Nov 2005.

[14] T. X. Brown, J. E. James, and A. Sethi. Jamming and sensing of encrypted wireless ad hoc networks. In *Proceedings of the ACM MobiHoc*, pages 120–130, 2006.

[15] D. Čabrić, S. Mishra, D. Willkomm, R. Brodersen, and A. Wolisz. A cognitive radio approach for usage of virtual unlicensed spectrum. *Proc. of the 14th IST Mobile and Wireless Communications Summit*, 2005.

[16] M. Cagalj, S. Capkun, and J.-P. Hubaux. Wormhole-based anti-jamming techniques in sensor networks. *IEEE Transactions on Mobile Computing*, 6(1):100–114, 2007.

[17] A. Chan, X. Liu, G. Noubir, and B. Thapa. Control channel jamming: resilience and identification of traitors. In *Proceedings of ISIT*, 2007.

[18] P. Chaporkar, K. Kar, X. Luo, and S. Sarkar. Throughput and fairness guarantees through maximal scheduling in wireless networks. *IEEE Transactions on Information Theory*, 54(2):572–594, 2008.

[19] T. Chen, H. Zhang, M. Katz, and Z. Zhou. Swarm intelligence based dynamic control channel assignment in CogMesh. In *Proc. of ICC*, pages 123–128, 2008.

[20] T. Chen, H. Zhang, G. Maggio, and I. Chlamtac. CogMesh: A cluster-based cognitive radio network. In *Proc. of DySPAN*, pages 168–178, 2007.

[21] J. T. Chiang and Y.-C. Hu. Cross-layer jamming detection and mitigation in wireless broadcast networks. In *Proceedings of the MobiCom*, pages 346–349, 2007.

[22] J. T. Chiang and Y.-C. Hu. Dynamic jamming mitigation for wireless broadcast networks. In *Proc. of INFOCOM*, pages 1211–1219, 2008.

[23] M. Dawande, P. Keskinocak, J. Swaminathan, and S. Tayur. On bipartite and multipartite clique problems. *Journal of Algorithms*, 41(2):388–403, 2001.

[24] M. Dawande, P. Keskinocak, and S. Tayur. On the biclique problem in bipartite graphs. Technical report, GSIA Working Paper, 1996-04. Carnegie Mellon University, 1996.

[25] J. Deng, R. Han, and S. Mishra. Intrusion tolerance and anti-traffic analysis strategies for wireless sensor networks. In *Proc. of IEEE CS Press Internal Conf. Dependable Systems and Networks*, pages 637 – 656, 2004.

[26] Y. Desmedt, R. Safavi-Naini, H. Wang, C. Charnes, and J. Pieprzyk. Broadcast anti-jamming systems. In *Proc. of the IEEE International Conference on Networks (ICON)*, pages 349 – 355, 1999.

[27] J. H. Dinitz and D. R. Stinson. A hill-climbing algorithm for the construction of one-factorizations and room squares. *SIAM J. Algebraic Discrete Methods*, 8(3):430–438, 1987.

[28] FCC. Spectrum Policy Task Force Report, 2002.

[29] FCC. OET Bulletin No.69–Longley-Rice Methodology for evaluating TV Coverage and Interference. *Federal Communications Commission*, 2004.

[30] S. Feng, H. Zheng, H. Wang, J. Liu, and P. Zhang. Preamble design for non-contiguous spectrum usage in cognitive radio networks. In *Proc. of WCNC*, 2009.

[31] I. . W. G. for WLAN. IEEE 802.15 for wireless local area networks (WLANs). http://www.ieee802.org/11/.

[32] I. . W. G. for WPAN. IEEE 802.15 for wireless personal area networks (W-PANs). http://www.ieee802.org/15/.

[33] G. Ganesan and Y. Li. Cooperative spectrum sensing in cognitive radio networks. In *Proc. of the DySpan*, pages 137–143, 2005.

[34] A. Ghasemi and E. Sousa. Collaborative spectrum sensing for opportunistic access in fading environments. In *Proc. of the DySpan*, pages 131–136, November 2005.

[35] A. Gupta, X. Lin, and R. Srikant. Low-complexity distributed scheduling algorithms for wireless networks. *IEEE/ACM Transactions on Networking (TON)*, 17(6):1846–1859, 2009.

[36] C. Han, J. Wang, Y. Yang, and S. Li. Addressing the control channel design problem: OFDM-based transform domain communication system in cognitive radio. *Computer Networks Journal*, 52(4):795–815, 2007.

[37] Y. C. Hu, D. Johnson, and A. Perrig. Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *Proc. of WMCSA*, 2002.

[38] H. Ishii and H. Kakugawa. A self-stabilizing algorithm for finding cliques in distributed systems. In *Proc. of the 21st IEEE Symposium on Reliable Distributed Systems (SRDS'02)*, pages 390–395, 2002.

[39] J. Jia, Q. Zhang, and X. Shen. HC-MAC: a hardware-constrained cognitive MAC for efficient spectrum management. *IEEE Journal on Selected Areas in Communications*, 26(1):106–117, 2008.

[40] D. Johnson, D. Maltz, and J. Broch. The dynamic source routing protocol for multihop wireless ad hoc networks. *Ad Hoc Networking (Addison-Wesley, 2001)*, pages 139–172, 2001.

[41] H. K. Kalita and A. Kar. Wireless sensor network security analysis. *International Journal of Next-Generation Networks (IJNGN)*, 1(1):1–10, 2009.

[42] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's Ad Hoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 1(2):293–315, 2003.

[43] P. Kondareddy, Y.R. Agrawal. Synchronized MAC protocol for multi-hop cognitive radio networks. In *Proc. of ICC*, pages 3198–3202, 2008.

[44] M. Krondorf, T. Liang, and G. Fettweis. On synchronization of opportunistic radio OFDM systems. In *Proc. of the IEEE Vehicular Technology Conference (VTC'08)*, pages 1686–1690, 2008.

[45] Y. W. Law, L. V. Hoesel, J. Doumen, P. Hartel, and P. Havinga. Energy efficient link-layer jamming attacks against wireless sensor network MAC protocols. In *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2005.

[46] L. Lazos and M. Krunz. Selective jamming/dropping insider attacks in wireless mesh networks. *IEEE Network*, 25(1):31–34, 2011.

[47] L. Lazos, S. Liu, and M. Krunz. Mitigating control-channel jamming attacks in multi-channel ad hoc networks. In *Proceedings of the 2nd ACM Conference on Wireless Network Security (WiSec)*, pages 169–180, 2009.

[48] L. Lazos, S. Liu, and M. Krunz. Mitigating control-channel jamming attacks in multi-channel ad hoc networks. In *Proc. of WiSec*, pages 169–180, 2009.

[49] L. Lazos, S. Liu, and M. Krunz. Spectrum opportunity-based control channel assignment in cognitive radio networks. In *Proceedings of SECON*, pages 135–143, 2009.

[50] M. Li, I. Koutsopoulos, and R. Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *Proceedings of the INFOCOM*, 2007.

[51] G. Lin and G. Noubir. On link-layer denial of service in data wireless LANs. *Journal on Wireless Communications and Mobile Computing*, 2004.

[52] A. Liu, P. Ning, H. Dai, Y. Liu, and C. Wang. Defending DSSS-based broadcast communication against insider jammers via delayed seed-disclosure. In *Proc. of the Annual Computer Security Applications Conference (ACSAC'10)*, 2010.

[53] X. Liu, G. Noubir, R. Sundaram, and S. Tan. SPREAD: Foiling smart jammers using multi-layer agility. In *Proceedings of the INFOCOM Mini Symposium*, 2007.

[54] Y. Liu, P. Ning, H. Dai, and A. Liu. Randomized differential DSSS: Jamming-resistant wireless broadcast communication. In *Proc. of INFOCOM*, 2010.

[55] Y. Liu, P. Ning, H. Dai, and A. Liu. Randomized differential DSSS: Jamming-resistant wireless broadcast communication. In *Proceedings of the INFOCOM*, 2010.

[56] J. M. McCune, E. Shi, A. Perrig, and M. K. Reiter. Detection of denial of message attacks on sensor network broadcasts. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2005.

[57] E. Mendelsohn and A. Rosa. One-factorizations of the complete graph-a survey. *Journal of Graph Theory*, 9(1):43–65, 1985.

[58] S. Mishra, A. Sahai, and R. Brodersen. Cooperative sensing among cognitive radios. In *Proc. of ICC*, June 2006.

[59] G. Noubir and G. Lin. Low-power DoS attacks in data wireless LANs and countermeasures. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(3):29–30, 2003.

[60] G. Noubir and G. Lin. Low power DoS attacks in data wireless LANs and countermeasures. In *Proceedings of the ACM MobiCom*, 2003.

[61] R. Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics Journal*, 131(3):651–654, 2003.

[62] C. E. Perkins and E. M. Royer. Ad hoc on-demand distance vector routing. In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90 – 100, 1999.

[63] R. Poisel. *Modern communications jamming principles and techniques*. Artech House on Demand, 2004.

[64] R. Poovendran and L. Lazos. A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks. *ACM/Springer Journal on Wireless Networks (WINET)*, 13(1):27–59, 2007.

[65] C. Popper, M. Strasser, and S. Capkun. Jamming-resistant broadcast communication without shared keys. In *Proc. of the USENIX Security Symposium*, 2009.

[66] C. Popper, M. Strasser, and S. Čapkun. Anti-jamming broadcast communication using uncoordinated spread spectrum techniques. *IEEE Journal on Selected Areas in Communication*, 28(5):703–715, 2010.

[67] A. Proaño and L. Lazos. Selective jamming attacks in wireless networks. In *Proceedings of ICC*, 2010.

[68] D. R. Raymond and S. F. Midkiff. Denial-of-service in wireless sensor networks: Attacks and defenses. *IEEE Pervasive Computing*, 7(1):74–81, 2008.

[69] N. Robertson, D. Sanders, P. Seymour, and R. Thomas. The four colour theorem. *Journal of Combinatorial Theory, Series B*, 70(1):2–44, 1997.

[70] H. A. B. Salameh, M. M. Krunz, and O. Younis. MAC protocol for opportunistic cognitive radio networks with soft guarantees. *IEEE Transactions on Mobile Computing*, 8(10):1339–1352, 2009.

[71] S. Sarkar and L. Tassiulas. End-to-end bandwidth guarantees through fair local spectrum share in wireless ad-hoc networks. *IEEE Transactions on Automatic Control*, 50(9):1246–1259, 2005.

[72] G. Sharma, C. Joo, and N. Shroff. Distributed scheduling schemes for throughput guarantees in wireless networks. In *Proc. of the 44th Annual Allerton Conference on Communications, Control, and Computing*, 2006.

[73] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt. *Spread Spectrum Communications Handbook*. McGraw-Hill, 2001.

[74] B. Sklar. *Digital Communications, Fundamentals and Applications*. Prentice-Hall, 2001.

[75] D. Slater, P. Tague, R. Poovendran, and B. Matt. A coding-theoretic approach for efficient message verification over unsecure channels. In *Proceedings of the ACM Conference on Wireless Security (WiSec)*, 2009.

[76] J. So and N. H. Vaidya. Multi-channel MAC for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver. In *Proceedings of the ACM MobiHoc*, pages 222–233, 2004.

[77] M. Strasser, C. Popper, and S. Capkun. Efficient uncoordinated FHSS anti-jamming communication. In *Proceedings of the ACM MobiHoc*, 2009.

[78] M. Strasser, C. Popper, S. Capkun, and M. Cagalj. Jamming-resistant key establishment using uncoordinated frequency hopping. In *Proceedings of IEEE Symposium on Security and Privacy*, 2008.

[79] K. Sun, P. Peng, P. Ning, and C. Wang. Secure distributed cluster formation in wireless sensor networks. In *Proc. of the Annual Computer Security Applications Conf. (ACSAC'06)*, 2006.

[80] P. Tague, M. Li, and R. Poovendran. Probabilistic mitigation of control channel jamming via random key distribution. In *Proceedings of PIRMC*, 2007.

[81] P. Tague, M. Li, and R. Poovendran. Mitigation of control channel jamming under node capture attacks. *IEEE Transactions on Mobile Computing*, 8(9):1221–1234, 2009.

[82] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, 2002.

[83] P. Tosic and G. Agha. Maximal clique-based distributed group formation for autonomous agent coalitions. In *Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, 2004.

[84] Y. Tseng, S. Ni, Y. Chen, and J. Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless Networks*, 8(2/3):153–167, 2002.

[85] W. Wallis. One-factorizations of complete graphs. *Contemporary Design Theory: A Collection of Surveys*, pages 692–731, 1992.

[86] W. Wallis. *One-factorizations*. Kluwer Academic Publishers, 1997.

[87] T. Weiss, A. Krohn, F. Capar, I. Martoyo, and F. Jondral. Synchronization algorithms and preamble concepts for spectrum pooling systems. In *Proc. of the IST Mobile and Wireless Telecommunications Summit*, 2003.

[88] A. Wood and J. Stankovic. Denial of service in sensor networks. *Computer*, 35(10):54–62, 2002.

[89] W. Xu, W. Trappe, and Y. Zhang. Anti-jamming timing channels for wireless networks. In *Proceedings of the 1st ACM Conference on Wireless Security (WiSec)*, 2008.

[90] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proc. of MobiHoc*, pages 46–57, 2005.

[91] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the ACM MobiHoc*, pages 46–57, 2005.

[92] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel surfing and spatial retreats: Defenses against wireless denial of service. In *Proceedings of Wireless Security Workshop (WiSe)*, 2004.

[93] J. Yu and P. Chong. A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys & Tutorials*, 7(1):32–48, 2005.

[94] J. Zhao, H. Zheng, and G.-H. Yang. Spectrum sharing through distributed coordination in dynamic spectrum access networks. *Wireless Communications and Mobile Computing Journal*, 7(9):1061–1075, 2007.