DEFENDING WIRELESS NETWORKS FROM RADIO INTERFERENCE ATTACKS

BY WENYUAN XU

A Dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Wade Trappe

and approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

October, 2007

ABSTRACT OF THE DISSERTATION

Defending Wireless Networks from Radio Interference Attacks

by Wenyuan Xu

Dissertation Director: Professor Wade Trappe

Wireless networks are built upon a shared medium that makes it easy for adversaries to conduct radio interference, or jamming attacks, which effectively cause a denial of service (DoS) of either transmission or reception functionalities. These attacks can be easily accomplished by an adversary by either bypassing MAC-layer protocols, or emitting a radio signal targeted at jamming a particular channel. In this thesis, we examine the issue of jamming wireless networks, and sensor networks in particular, by studying both the attack and defense side of the problem. On the attack side, we present four different jamming attack models that can be used by an adversary to disable the operation of a wireless network, and evaluate their effectiveness in terms of how each method affects the ability of a wireless node to send and receive packets. In order to cope with the problem of jamming, we discuss a two-phase strategy involving the diagnosis of the attack, followed by a suitable defense strategy. For detection, we show that single measurement statistics are not enough to reliably classify the presence of a jamming attack, and propose multimodal detection methods. To cope with jamming, we propose a technique, channel surfing, which involves evading the interferer in the spectral domain. Several different channel surfing models are presented, and we evaluate their effectiveness using a testbed of MICA2 motes. Beyond channel surfing, we overview a second defense strategy whereby it is possible to establish a low data rate jamming resistant communication channel by modulating the interarrival times between jammed packets.

Acknowledgements

This dissertation would not have been possible if it were not for the guidance, support, help and friendship of many people.

First and foremost, I would like to express my utmost gratitude to my advisor Prof. Wade Trappe, for his inspiring guidance, constant encouragement, and complete support during the course of my studies. I am also greatly indebted to my co-advisor Prof. Yanyong Zhang, who has been a constant source of guidance and encouragement over the past year. In particular, I thank them for providing good models of intellectual rigor and innovative scholarship, and for helping me improve my presentation and writing skills. They have not only been good advisors, mentors, but also have been good friends.

I am grateful to many professors at Rutgers who have helped me in my professional development. In particular, Prof. Dipankar Raychaudhuri, Prof. Marco Gruteser and Dr. Sanjoy Paul have given me valuable comments that have helped improve the quality of my research. I would like to especially acknowledge Ivan Seskar, who has helped me acquire the skills needed to conduct many of my experiments. His deep knowledge of wireless systems, at so many different layers, has both been helpful and inspiring.

I would like to express my gratitude to many of my colleagues. First, I must acknowledge all of the members of Prof. Trappe's research group. Unfortunately, there are too many to list here, but they know who they are! Beyond my immediate research group, Zhibin Wu, Rouheng Liu, Xiangpeng Jin, Suli Zhao, Sumathi Gopal, Sachin Ganu, and many others have help me along the way with stimulating discussions and I would like to thank them.

Outside of Rutgers, I would like to extend my thanks to William Horne and other members of the Hewlett-Packard Trusted Systems Lab. The summer that I spent with them helped expand my horizons as a researcher.

Finally, to my husband Yi Jin, my mother, and my brother, thanks for always being there for me, and helping to keep me going through both the good and bad times during my graduate studies.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Wireless networks are progressively becoming more affordable, and consequently are being deployed in a variety of different modalities, ranging from wireless local area networks to mesh and sensor networks. For example, WiFi (IEEE 802.11) is widely used both for residential and enterprise settings to enable users to access the Internet. WiMAX (IEEE 801.16) provides users high-speed broadband network access in a metropolitan area. Further, networks are being deployed in a wide range of settings to monitor and collect data. As people increasingly rely on these wireless technologies to exchange crucial information, being able to assure their security and trustworthiness is an issue of critical importance.

The essential security requirements include confidentiality, integrity, authentication and availability. Confidentiality requires that the data exchanged is only accessible to those authorized users, usually the sender and the receiver involved in communication. Integrity means the message sent has not been altered. For instance, the message could be corrupted by transmission errors. Authentication procedures verify the identity of the sender. Threats to those security services, such as confidentiality, integrity, and authentication in wireless networks, may be addressed through appropriately designed network security architectures [2, 26, 27, 31, 54, 60, 74, 82], which are essentially application of traditional cryptographic functions, to the wireless domain. Wireless networks, however, are susceptible to adversarial and non-adversarial threats that will breach the *availability* of wireless networks. Those threats are not able to be adequately addressed via cryptographic methods.

Assuring the availability of wireless networks is complicated by challenges that arise from the unpredictable nature of signal propagation. Furthermore, the shared nature of the wireless medium, combined with the commodity nature of wireless technologies and an increasingly sophisticated user-base, allows wireless networks to be easily monitored and broadcast on. At the most basic level, adversaries may easily observe communications between wireless devices,

and just as easily launch simple denial of service attacks against wireless networks by injecting false messages. Traditionally, denial of service is concerned with filling user-domain and kernel-domain buffers [28]. However, in the wireless domain, the availability of the wireless network may be subverted quite easily through radio interference.

In this dissertation, we focus on address the problem of ensuring the availability of wireless networks in the presence of radio interference. Such interference may be either incidental or intentional– corresponding to the accidental scenario where many transmitters share the same "radio" environment (e.g. the same 802.11 channel may be used by many users in an apartment building), or may involve one or more intentional adversaries seeking to disrupt the continuity of network services by actively transmitting or occupying the wireless channel. A general characteristic of such Radio Frequency (RF) interference scenarios is that the wireless medium is blocked and other wireless devices are prevented from communicating and/or receiving.

1.2 Problem Overview

We set the stage for the problem by starting with a parable. Suppose Alice and Bob are socializing with each other at a party and, suddenly, the malicious Mr. X walks up. Without any regard for proper social etiquette, he interrupts them and begins to take over the conversation. Each time Alice tries to talk, Mr. X interrupts her and tells an inane story. Bob, likewise, doesn't fare any better. Alice and Bob both wait a polite amount of time in order to give Mr. X an opportunity to remedy his behavior. However, after some time, it becomes clear that Mr. X will not give in and that our two heroes are destined to have a poor reunion and regret ever attending the party.

The story of the social party is a simple, motivating example for the problem of wireless radio interference we study in this dissertation. In the case of wireless communication, Alice and Bob correspond to two communicating nodes A and B, while Mr. X corresponds to an adversarial interferer X. The adversary X, who may or may not be intentionally trying to disrupt communication, may interfere with A and B's ability to communicate by either ignoring MAC-layer protocols (e.g. perhaps X does not know the proper MAC-layer etiquette or perhaps he actively chooses to ignore MAC protocols), or by emitting a signal of sufficient energy on

the channel used by A and B.

Throughout this thesis, we shall refer to the entity that performs RF-interference as the *jammer* or the *interferer*. RF interference, whether intentional or not, can be launched against wireless networks easily by various means. From the adversary point of view, he/she can build MAC-layer jammers by having the wireless devices simply disregard the medium access protocol. For example, in 802.11 networks, an adversary may employ a powerful device driver to bypass card firmware and repeatedly send out packets. As a consequence, all devices within the radio range of X will think that the channel is occupied and will defer transmission of data. Similarly, at the PHY-layer, an adversary may use any device (e.g. waveform generator) capable of emitting energy in the frequency band corresponding to the channel A and B are communicating on. Furthermore, though not necessary legal, many off-the-shelf jamming devices are available for different purposes. Cell phone jamming units can be purchased and installed in classrooms or theaters to block cell phone reception signals. Beyond those intentional jammers, unintentional RF-interference can be witnessed in our daily life as wireless networks have become increasingly popular. Various wireless devices operate on "open spectrum". As a result, different devices may have overlapping operational frequency bands, while their spectrum access protocols are not designed to cooperate with each other to minimize inter-protocol RF interference. It is well-known that Bluetooth devices, cordless phone and microwaves share spectrum with 802.11b/g, and hence can be used to interfere with an 802.11b/g network. [14, 15, 18, 48]

Jamming attacks on the physical layer have been known by the communications and radar community for some time and there are numerous texts, such as [61, 64, 71], which discuss issues associated with these attacks. Typically, in the context of those communication systems, the objective of the jammer is to deny the reception of communications at the receiver. In these systems, jamming is usually addressed through spreading techniques, whereby resilience to interference is achieved by transmitting information using a bandwidth much larger than its required minimum bandwidth. One of the well known spread spectrum techniques is frequency hopping, whereby information is transmitted across multiple frequencies sequentially according to a predefined frequency-hopping sequence. By doing so, the radio interference will "hit" limited number of frequency, and only a portion of frequency will be blocked instead of the

whole frequency. Thus, the chances of recovering the original messages are increased.

The traditional approach to coping with radio interference implies that more expensive transceivers are used and, with the exception of some military systems, most commodity sensor and wireless networks do not employ sufficiently strong spreading techniques to survive jamming or to achieve multiple access. Instead, for reasons of cost, systems like the Berkeley MICA2 [3], the Zigbee (e.g. MICAZ), 802.15.4, and even 802.11 are based upon a carrier-sensing approach to multiple access. Because of their use of carrier sensing for medium access control, these systems are extremely susceptible to simple jamming. Such MAC and PHY-layer security threats for wireless networks have been revisited recently by the Australian CERT [6], and represent a critical vulnerability for wireless networks. Throughout this thesis, we shall examine the problem of radio interference for wireless networks, with a particular emphasis placed on sensor networks, and propose mechanisms for detecting and coping with radio interference.

Much effort has been dedicated to examining the problem of radio interference via theoretical tools or via simulation. However, the majority of that work remains untested in a real deployment. One of the objectives for this dissertation is to field-test the real effectiveness of radio interference, issues related to detect radio interference, and to evaluate the ability to cope with radio interference on sufficiently large wireless networks. In real wireless networks, the actual effect of radio interference is often quite different from idealized mathematical models. An account of discrepancies between radio interference in real systems and expectations from modeling, thus, is valuable. To our best knowledge, extensive testing on large networks has not been undertaken, this dissertation presents one of the first efforts along these lines. Additionally, this dissertation presents defense mechanisms that have been implemented in real systems, and hence are capable of being deployed.

1.3 Overview of the Thesis

In this dissertation, we examine the issues of jamming wireless networks from both the attack and defense point of view. In particular, we propose several different jamming attack models, and evaluate those jammers on the MICA2 platform. We further study issues related to

detecting jamming and coping with jamming through a spectral evasion strategy on a 30-node MICA2 testbed. A MICA2 node is a small sensor device with radio and operating system on it. These devices each have a $902 - 928MHz$ Chipcon CC1000 radio, and the operating system running on each mote is TinyOS version 1.1.x.

The organization of the dissertation is as follows:

In Chapter 2, we briefly overview the problem of radio interference for various types of wireless networks. In particular, we provide examples of how easily RF-interference can disrupt wireless communications, and give a high-level overview of a general strategy to overcome interference, which consists of a multi-phased approach involving interference detection and then the use of an evasion strategy.

We then turn to examine the radio interference problem in ad hoc networks. In Chapter 3, we present different jamming attack strategies that might be used against wireless networks and measure their effectiveness by implementing all jamming strategies on a testbed of MICA2 motes.

Later, in Chapter 4, we examine methods that can be employed by the wireless network in order to detect the presence of jamming. We illustrate that basic statistics, like signal strength, alone are not sufficient for classifying the presence of a jammer, and more advanced detection methods are needed. We then show that multi-modal detection strategies, which involve the use of multiple statistics, can effectively identify jamming and discriminate jamming from accidental or legitimate causes of poor link quality.

In Chapter 5, we examine a strategy for coping with jamming. This strategy involves the wireless devices in the network adapt their channel assignments to restore network connectivity in the presence of interference. In particular, we explore two different approaches to channel surfing: coordinated channel switching, where the entire sensor network adjusts its channel; and spectral multiplexing, where nodes in a jammed region switch channels and nodes on the boundary of a jammed region act as radio relays between different spectral zones.

In Chapter 6, we provide a brief history of jamming, and related literature.

Finally, we conclude the thesis with a summary of the research and lessons learned, and outline directions for future work in Chapter 7. In particular, we present an overview of some

additional strategies for coping with interference and jamming. We examine the ability of legitimate devices to overcome jamming by adjusting the transmission power levels, and present a potentially powerful defense mechanism that involves the creation of a low data-rate, jamming-resistant channel by modulating the inter-arrival time between packets sent from a sender to a receiver.

Chapter 2

A Brief Survey of Jamming and Defense Strategies

In this chapter, we will briefly overview the problem of radio interference by providing examples of how easily RF-interference can disrupt wireless communications in various wireless networks. We will then provide a high-level overview of a general strategy to overcome interference, which consists of a multi-phased approach involving interference detection (see Chapter 4) and then the use of an *evasion* strategy. Although, in this chapter, we will outline two different evasion strategies, channel surfing and spatial retreats, we will focus our attention only on a more detailed analysis of channel surfing in Chapter 5.

We begin in subsection 2.1 by presenting case studies on the RF-interference problem. We introduce the three different wireless network scenarios that we will study in this section in subsection 2.2. Following the setup of the problem, we present *channel surfing*, our first defense against MAC/PHY-layer denial of service attacks in subsection 2.3. Channel surfing involves valid participants changing the channel they are communicating on when a denial of service attack occurs. In subsection 2.4, we examine *spatial retreats*, which involves legitimate network devices moving away from the adversary.

2.1 Interference Case Studies

We now look at three different case studies that examine the performance of a wireless network in the presence of interference. The first case study demonstrates an adversarial case of interference involving a jammer that either bypasses the MAC-layer or uses a waveform generator to disrupt communications, the second case study examines the impact of cross-channel interference, while the third involves examining the effect of interference that arises due to natural congestion in carrier sense multiple access based (CSMA-based) networks (e.g. 802.11). Our case studies show that both malicious jammers and MAC compliant network nodes can cause interference within wireless communications.

In most forms of wireless medium access control, there are rules governing who can transmit at which time. For example, one popular class of medium access control protocols for wireless devices are those based on carrier sense multiple access (CSMA). CSMA is employed in Berkeley motes as well as in both infrastructure and infrastructureless (ad hoc) 802.11 networks. In this section, we examine the impact that various interferers have on networks employing CSMA. The metrics we used to quantify the impact of the interferers are, (1) carrier sensing time, i.e. the amount of time a device spent in sensing the channel before transmission; (2) throughput, i.e. the amount of data bits that have been delivered from a network node to another; and (3) packet delivery ratio (PDR), i,e. the ratio of the number of data packets received by the destination node over the number of data packets sent by the source node.

2.1.1 Non-MAC-compliant Interferer Case Study

In this section, we study the impact that a Non-MAC-compliant interferer has on communication. In particular, we examine the impact the interferer has on carrier sensing time as well as packet delivery ratio (PDR).

During normal operation of CSMA, when A tries to transmit a packet, it will continually sense the channel until it detects the channel is idle, after which it will wait an extra amount of time (known as the propagation delay) in order to guarantee the channel is clear. Then, if RTS/CTS is used it will send the RTS packet, or otherwise will send the data packet. Suppose the adversary X is continually blasting on a channel and that A attempts to transmit a packet. Then, since X has control of the channel, A will not pass carrier-sensing, and A may time-out or hang in the carrier-sensing phase. As a result, no packets from A can be sent out.

To validate our analysis carrier sensing time and PDR we performed a set of experiment using MICA2 motes. Each mote had a ChipCon CC1000 RF transceiver and used TinyOS 1.1.1 as the operating system, which used a fixed threshold for determining idleness. To build the jammer X, we disabled the back off operations to bypass the MAC protocol.

Carrier Sensing time: To validate the effect of jammers on the carrier sensing time in a real wireless network, we performed an experiment using two Motes, X and A. Here, Mote A corresponds to a network node trying to send out a 33-byte packet every 100ms, and which

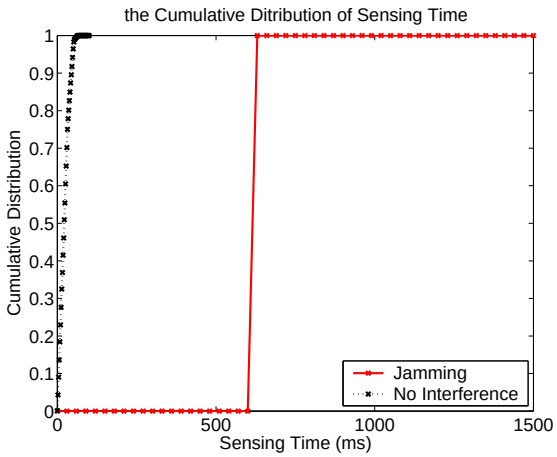


Figure 2.1: The cumulative distribution for the carrier sensing times measured using MICA2 Motes.

measures the sensing time while doing so. Mote X, the jammer, continuously sends out random bits without a time gap in between transmissions. Additionally, we measured the sensing time without any interference, i.e. X does not send any bits.

Fig. 2.1 depicts the cumulative distribution of the sensing time for those two scenarios. Fig. 2.1 shows that most of the carrier sensing time in the non-interference scenario is less than $100ms$, while the cumulative distribution of the jamming scenario jumps at the point where the sensing time equals to 640ms. This is caused by a timeout we added to the TinyOS. In our experiment, if the device does not start to send the packet within 640ms after the packet was passed to the MAC-layer from application layer, a timeout will occur, the packet will be discarded, and its sensing time will be counted as 640ms.

PDR: In this set of experiments, we have a sender A, receiver B and jammer X. Rather than constantly transmitting random bits, the jammer X here listens to the channel, and upon detection of preambles of a data packet, it immediately blasts on the channel by sending a random 2-byte jamming packet, causing the data packet garbled when it arrives at the receiver. We measured the PDR at the receiver B by calculating the ratio of the number of packets that have been correctly received with respect to the total number of packets that are expected to send. The number of expected packets can be obtained through tracking the sequence number of each packet. If no packets are received, the PDR is defined to be 0.

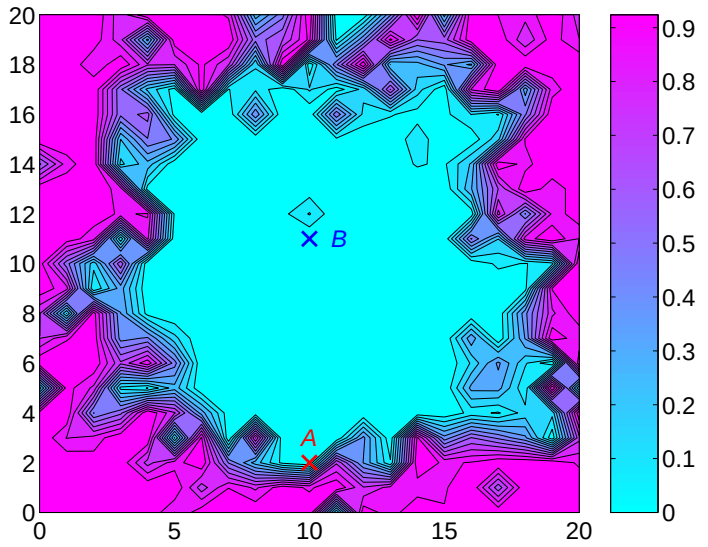
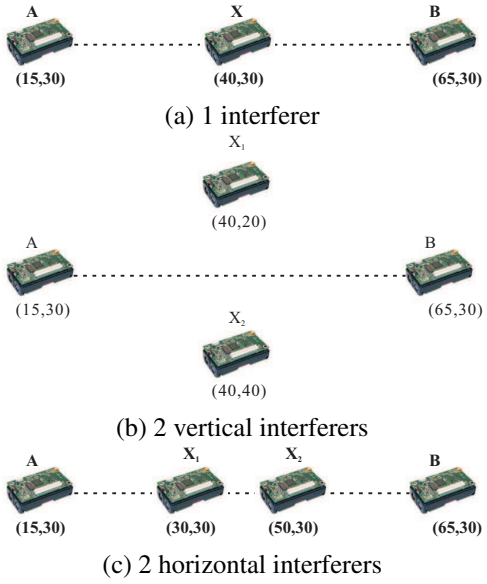


Figure 2.2: The PDR contours as measured for varied jammer locations on a 20X20 square feet grid with the sender's and the receiver's locations fixed. The blue cross represents the receiver, and the red cross is the sender. The color goes from light blue (0: no packets arrived at the receiver correctly) to purple (all packets arrived correctly). We note that the figure is best viewed in colored display.

The interference level caused by the jammer X is governed by a number of factors. The most obvious one is the relative location of the jammer and the receiver. We have studied the degradation of the PDR caused by the jammer X as a function of the jammer's location in a 20x20 ft. square grid. We have evaluated a total of 439 points in space by varying the jammer's location while keeping the sender's and the receiver's locations fixed.

We illustrated the degradation of the PDR by means of a color map in Figure 2.2. The resulting PDR contours, i.e. the lines that connect points on the grid with the same PDR levels, encompass the sender and the receiver which are located at $(10, 2)$ and $(10, 11)$ respectively. The obstruction of the PDR is roughly determined by the separation distance between the jammer and the receiver with a steep threshold occurring between 7 and 9 feet away from the receiver. When the jammer X is closer than 7 feet to the receiver, almost all the communication is corrupted. Due to the type of the jammer, the PDR is also sensitive to the separation between the sender and the jammer. In theory, the contours of the PDR between the sender and the receiver are circles centered at the receiver. However, due to the radio irregularity in an indoor environment which is not accounted for in theoretical model, the PDR contours are



(a) 1 interferer

(b) 2 vertical interferers

(c) 2 horizontal interferers

Figure 2.3: The locations of the communicators (A and B) and interferers (X, X_1 and X_2). The unit of distances is centimeter.

irregular. There is also a special case of greater than zero PDR when the jammer is in the immediate proximity of the receiver, appearing inside the diamond-shaped contours around $(10, 12)$, north to the receiver.

2.1.2 Cross-channel Interference Case Study

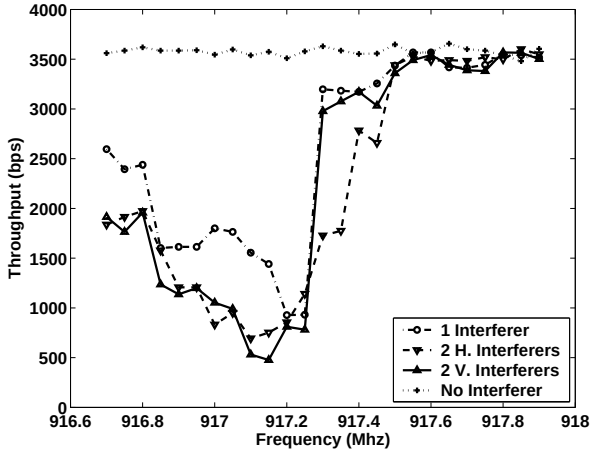
In this discussion we examine the issue of channel selection and the impact it can have on communications. Unlike the interferers studied in the previous case study, where the jammers do not follow MAC rules and can completely take the channel, the interferers discussed in this sub-section can be regular network nodes that follow a proper MAC protocol.

In order to demonstrate how channel selection can interfere with network communications, we conducted a set of experiments using Berkeley motes. In these experiments, two motes act as the communicator and receiver, denoted by A and B. A continuously sends out 31-byte packets to B, resulting in a throughput of 3.6Kbps. We then placed interferers or jammers in different locations. In the first set of experiments, we used motes as interferers. We tried three interferer scenarios, which are illustrated in Figures 2.3. These interferers also continuously

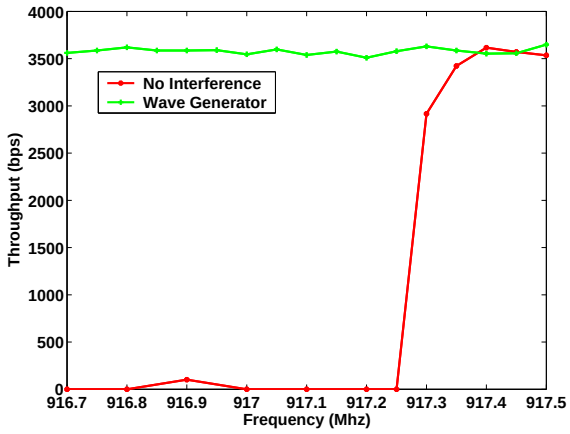
send out packets of the same size. The interferers completely followed the default MAC protocol of Berkeley motes. All the motes transmit at the same power level. The default frequency of the motes was 916.7MHz. The results for this set of experiments are summarized in Figure 2.4 (a). When all the motes transmit at the default frequency, the measured throughput between A and B significantly drops compared to the scenario with no interferers. Due to the fact that all devices follow the MAC protocol, the throughput did not become zero. We then incremented the transmission/reception frequency of the communicator and receiver by 50KHz. As the frequency gap between the communicators and the interferers increases, but before it reaches a threshold, the measured throughput worsens. This is because the transmissions still interfere with each other, yet the MAC protocol is not able to coordinate transmissions across different frequencies, resulting in a much higher collision rate and a lower throughput. Finally, when the communicators increase their frequency to (or above) 917.5MHz, they are no longer interfered with.

The lessons we learned here is that when there are nodes working on multiple channels, it is very important for each node to choose an interference-free channel, i.e. orthogonal channel, to avoid cross-channel interference. Although the specifications for the radio employed in Berkeley motes state that a channel separation of 150kHz is recommended in order to prevent cross-channel interference. From our experiment, we have found that 800kHz is a safer value for channel separation in order to maintain *effective network-layer orthogonality*, resulting in 32 orthogonal channels on MICA2 platform.

To understand how the jammer affects the network communication when the interfering signal is not centered on the channel that the network is operating on, we used a waveform generator as the jammer interfering with the two communicators. The position setup is the same as depicted in Figure 2.3(a), where X represents the location of waveform generator. The waveform generator continuously emitted a narrow AM signal at 916.7MHz frequency and with an amplitude of -10dBm. Unlike the interferers in the above experiment, the jammer does not follow MAC rules and can completely take the channel so that the communicators do not have a chance to transmit. The results are shown in Figure 2.4(b). Before the communicators move out of the spectral interference range, the measured throughput is 0. As soon as the communicators move to (or above) 917.4MHz, they can transmit without any interference. The reason for a



(a)



(b)

Figure 2.4: Experimental results indicating throughput versus channel assignment for (a) MAC-compliant Mote, (b) Waveform generator continuously emitting an AM signal.

narrower gap in this scenario compared to the mote interferer cases is that the spectral width of the waveform generator's signal is narrower than the spectrum of a Berkeley motes' signal.

2.1.3 Congestion Case Study

When the adversary X is continually blasting on a channel, if A attempts to transmit a packet, A's carrier-sensing time will be infinity or some time-out value. This time-out, however, could have occurred for legitimate reasons, such as congestion. Congestion can cause unintentional radio interference as well. To understand the relationship between congestion and interference,

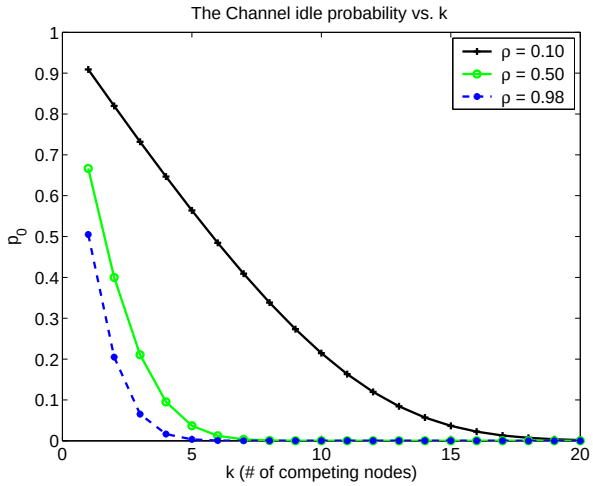


Figure 2.5: The steady-state probability that the channel is idle as the number of competing nodes increases using finite customer queue.

we employed two different approaches to examine how congestion effects network communication as unintentional interference sources. We start by looking at a theoretical channel occupancy model, and then present our empirically study.

The theoretical model: For the theoretical channel occupancy model, we will employ a simplified model for CSMA as well as some simplifying assumptions about the underlying network traffic. We assume that there are a finite set of radio sources that each acts as an independent Poisson source with a packet generation rate of λ packets/sec. Further, we assume that transmission of packets is completed at an average rate of μ according to an exponential distribution[1]. λ represents the traffic load. For example, λ is larger in the congested scenario than a light-weighted traffic scenario. μ depends on the size of a packet and channel capacity.

Under these assumptions, we may model the channel as a two-state Markov model, where state 0 is the Channel-Idle state and state 1 is the Channel-Occupied state. The two-state model for the channel is simply an $M/M/1/k/k$ queue, where we have finite number of customers (k customers) and a single server. In our case, the single server is the channel, and the customers are the network nodes that potentially compete with each other for the channel. We may use

[1]We note that the validity of our model depends on λ/μ, and that the approximations made become increasingly accurate as $\lambda/\mu \rightarrow 0$, c.f. [36].

queuing theory to derive $p_0 = P(\text{state-0})$, the steady-state probability that the channel is idle at an arbitrary time [35]. Let $\rho = \lambda/\mu$, then at steady state $p_0 = 1/\sum_{i=0}^{k} \frac{k!\rho^i}{(k-i)!}$. For a given ρ, p_0 is a function of k, the number of competing nodes. We plot $p_0(k)$ for three different levels of ρ in Figure 2.5. In particular, $\rho = 0.1$ represents a light weighted traffic load scenario, where each node tries to occupy one tenth of the channel capacity; $\rho = 0.98$ corresponds to a heavily loaded traffic scenario, where every node tries to saturate the channel. The plots are inline with the intuition, i.e. as the number of competing nodes increases, the channel idle probability decreases. The higher the packet generation rate per node, the faster the channel gets saturated. Therefore, in a congested scenario, a node has to wait longer time before it can access the channel and start to transmit.

Now, let's look at the carrier sensing time. Suppose that A senses the channel at an arbitrary time t. We are interested in the amount of time D that A must, under CSMA, wait before it starts transmission. A will sense the channel until it is idle. Under popular implementations of CSMA, when the channel becomes idle, node A will continue sensing the channel for an additional τ time before transmitting in order to ensure distant transmissions have arrived at A[2]. If the channel becomes busy during that time, then A declares the channel busy and continues sensing until it experiences an idle period that lasts τ units of time.

In order to capture the distribution for D, we will introduce a few auxiliary variables. Let N denote the number of times we visit state 0 before witnessing an idle period of at least τ duration. Also, let us define the random variable $T_{0,k}$ to be the duration in state 0 for the k-th visit to state 0, and similarly $T_{1,k}$ to be the duration in state 1 for the k-th visit to state 1. Observe that, if we are at state 0, then to exit CSMA requires $T_{0,k} \geq \tau$, and hence we must introduce $q = P(T_{0,k} \geq \tau) = e^{-\lambda\tau}$. We may now look at the chain of events, as depicted in Figure 2.6. If the channel was idle when we started, then we enter at the left-most state 0, whereas if the channel was occupied when we entered, then we enter at the left-most state 1.

[2]Typically, the propagation delay τ is on the order of 50μseconds.

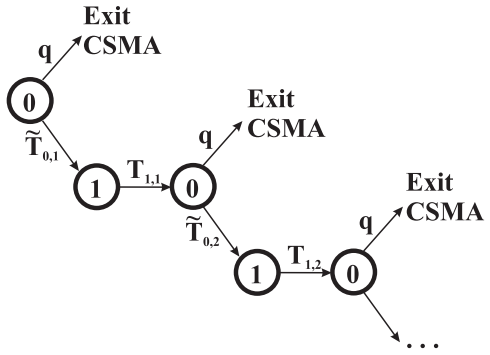


Figure 2.6: The events leading to exiting CSMA, and the corresponding probability parameters used in the derivation.

The probability density function for D can be shown to be

$$f_D(d) = p_0 \sum_{k=1}^{\infty} P(N=k) f_{0,k}(d) \tag{2.1}$$

$$+(1-p_0)\left(f_{T_{1,1}} + \sum_{k=1}^{\infty} P(N=k) f_{0,k}(d)\right) \tag{2.2}$$

$$= (1-p_0) f_{T_{1,1}} + \sum_{k=1}^{\infty} P(N=k) f_{0,k}(d) \tag{2.3}$$

Here N is geometric with $P(N=k) = (1-q)^{k-1}q$. The distributions $f_{0,k}(d)$ are the pdfs describing the duration contributed by exiting during the k-th visit to state 0 when we start at state 0. For example, if we exit after the first visit to state 0, then the only contribution comes from the propagation delay τ, that is, $f_{0,1}(d) = \delta(\tau)$, the point mass at $d = \tau$. As another example, let us look at the time, Y, contributed when exiting at the second visit to state 0. Since we did not exit during the first visit to state 0, Y is composed of time contributed by $T_{0,1}$ that is strictly less than τ. We will call this conditional random variable $\tilde{T}_{0,1}$. Next, Y also consists of $T_{1,1}$, and finally the τ needed to exit CSMA. Hence $Y = \tilde{T}_{0,1} + T_{1,1} + \tau$, and thus

$$f_{0,2}(d) = f_{\tilde{T}_0} * f_{T_1} * \delta(\tau) = f_{0,1} * f_{\tilde{T}_0} * f_{T_1} \tag{2.4}$$

where $*$ denotes convolution and arises due to independence between the random variables involved, and f_{T_j} is the distribution for occupancy time for any state j. Similar recursive representations can be derived for $f_{0,k} = f_{0,k-1} * f_{\tilde{T}_0} * f_{T_1}$.

From this theoretical pdf, we may calculate the mean of carrier sensing time $E[D] = \int_0^\infty f_D(x)xdx$. Now let's examine the trend of $E[D]$. When k the number of competing

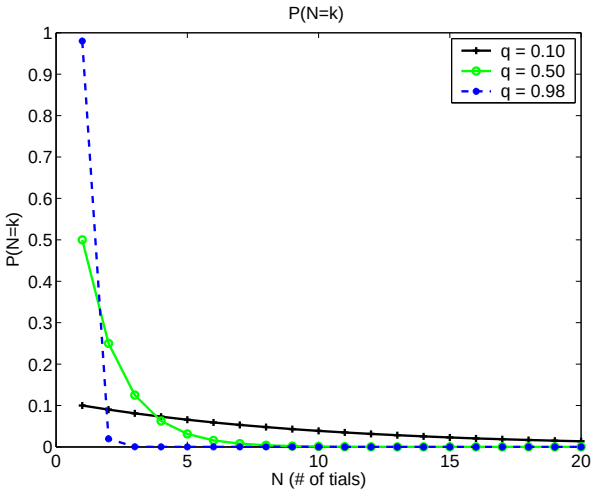
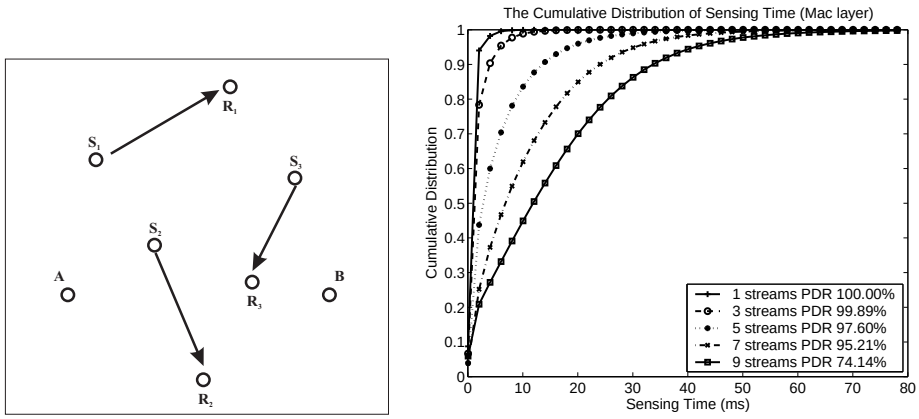


Figure 2.7: The probability that a node will exit CSMA at the kth' trial.

nodes increases and λ the aggregated traffic load increase, the probability q that a node can exit CSMA at a kth stage decreases. Thus, the shape of $P(N = k) = (1 - q)^{k-1}q$ over k becomes flatter with longer tail, as show in Figure 2.7. The flatter shape of $P(N = k)$ over n indicates that the $E[D]$ becomes larger as the traffic load increase. Although the regular network nodes is CSMA compliant, the increasing density of the nodes plus the increasing traffic load together will lead to higher carrier sensing time, and thus higher interference.

The experimental study: A second approach involves each network device collecting statistics regarding the amount of time D that a device must wait before it can start transmission during normal, or even somewhat congested, network conditions. After collecting enough statistics, we calculate the distribution $f_D(d)$ which describes the amount of time spent in sensing before the channel becomes idle during various network traffic conditions, including light-weighted and congested scenarios.

In order to measure the distribution $f_D(d)$, we carried out several experimental studies using the 802.11 extensions to the ns-2 simulator. We modified ns-2 by disabling the MAC layer retransmission, so that we could focus our investigation on the channel sensing behavior. In our experiments we have two nodes, A and B. Once every 19 msecs, node A senses the channel by trying to send out a beacon to node B. We obtained the MAC-layer delay time D by calculating the difference between the time when beacon packets reach MAC-layer and



(a) (b)

Figure 2.8: The MAC-layer sensing time experiment: (a) basic underlying experimental setup, (b) cumulative distributions of D for different traffic scenarios and the corresponding packet delivery ratio.

the time when the MAC successfully senses the channel as idle and sends out RTS. In order to capture the statistical behavior of the delay time, we calculated the cumulative distribution of the delay time for several scenarios involving different levels of background traffic loads. As shown in Figure 2.8 (a), we introduced several streams (from sender S_i to receiver R_i) that are within the radio range of A and B in order to increase the interfering traffic. Each stream's traffic was chosen to represent an MPEG-4 video stream suitable for a wireless video application. We used traffic statistics corresponding to the movie Star Wars IV [20], where each sender transmitted packets with the packet size governed by an exponential distribution with a mean size of 268 bytes, and the packet inter-arrival times following an exponential distribution with mean 40msecs, resulting in each stream having an average traffic rate of 53.6Kbps. The corresponding cumulative distributions of D are shown in Figure 2.8 (b). These observations can be explained as follows. When there are only a few streams, there are few nodes competing for the channel, and node A can get the channel quickly with high probability. As the number of streams increases, the competition for the channel becomes more intense, thus taking longer for A to acquire the channel. This observation agrees with the result we get from our theoretical channel occupancy model.

From this figure, we can observe that the cumulative distribution of carrier sensing time reaches its saturation point faster when the number of streams is small. For instance, in the 1

stream scenario, 90% of the time, a node waits less than 5ms before it starts to transmit, while in 9 stream scenario, only 20% the carrier sensing times are less than 5ms. Thus, the interference condition is more severe when the number of streams is bigger.

Additionally, we can observe that as the number of streams increases the average packet delivery ratio decreases. Packet delivery ratio is another important parameter to quantify the network interference condition. The lower the packet delivery ratio is, the more severe the interference is. For instance, when the number of streams is 9, the average packet delivery ratio is 74.1% and corresponds to a poor quality of service for each application.

2.2 Defense: Detection and Evasion

In order to cope with interference, whether intentional or accidental, we have proposed a two-phase strategy involving the diagnosis of the radio interference/jamming, followed by a suitable defense strategy. We highlight the challenges associated with detecting jamming, and refer the reader to Chapter 4 for more discussion on detection. Coping with the problem of overcoming interference is very challenging, as well. At one level, if the interferer is malicious and relatively powerful, then the only viable strategy for defense is to employ sophisticated physical layer methods with large anti-jam margins. On the other hand, for interference scenarios that arise from accidental interference or only modest levels of jamming, there are other defense methods that don't involve costly physical layer methods. Although it might seem that there is nothing that can be done since assuming that an adversary can continually blast on a channel grants the adversary considerable power, we nonetheless take inspiration for our work from the famous Chinese war strategy book, *Thirty-Six Stratagems*:

> *He who can't defeat his enemy should retreat.*

Translating this philosophy into the wireless domain, we may propose that wireless devices employ spectral and spatial evasion strategies in order to protect against interference.

Although there are many different scenarios where jamming may take place, we will focus on three basic classes of wireless networks, as depicted in Figure 2.9:

1. **Two-Party Radio Communication:** The two-party scenario is the baseline case, in which A and B communicate with each other on a specific channel. As long as interferer

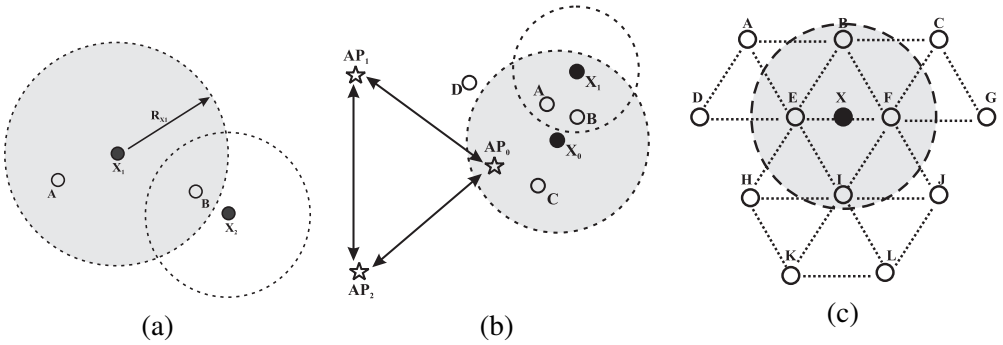


Figure 2.9: The three wireless communication scenarios studied in this paper: (a) two-party radio communication, (b) infrastructured wireless networks, and (c) ad hoc wireless networks. The adversary is depicted by X.

X is close enough to either A or B, its transmission will interfere with the transmission and reception of packets by A and B.

2. **Infrastructured Wireless Networks:** Infrastructured wireless networks, such as cellular networks or wireless local area networks (WLANs), consist of two main types of devices: access points and mobile devices. Access points are connected to each other via a separate, wired infrastructure. Mobile devices communicate via the access point in order to communicate with each other or the Internet. The presence of an interferer, such as X_0 or X_1, might make it impossible for nodes to communicate with their access point.

3. **Mobile Ad Hoc Wireless Networks:** Ad hoc networks involve wireless devices that establish opportunistic connections with each other in order to form a communication network. Typically, ad hoc networks employ multi-hop routing protocols in order to deliver data from one network node to another. The presence of an interferer may bring down whole regions of the network.

It should be noted that in all three cases, the interferer could have limited interference range, thereby affecting only one of two communicating participants. For example, in Figure 2.9 (a), interferer X_2 is located to the right of B and blocks B from being able to acquire the communication channel while A might not be able to detect X_2.

2.3 Channel Surfing Overview

The first escape strategy that we present is channel surfing. Typically, when radio devices communicate they operate on a single channel. When an interferer comes in range and blocks the use of a specific channel, it is natural to migrate to another channel. The idea of channel surfing is motivated by a common physical layer technique known as frequency hopping. We assume throughout this section that the interferer/jammer blasts on a single channel at a time, and that the interferer/jammer cannot pretend to be a valid member of the network (i.e. the jammer does not hold any authentication keys used by the network devices). In this section, we examine the channel surfing strategies for three basic classes of wireless networks listed in previous section. We will explore the case of an ad hoc network in more detail in Chapter 5.

2.3.1 Two-Party Radio Communication

Consider the radio scenario depicted in Figure 2.9(a). In this scenario, jammer X_1 or X_2 has disrupted communication between A and B. We desire both A and B to change to a new channel in order to avoid X's interference. Using some detection techniques discussed in Chapter 4, once A and B have detected jamming, they will change to a new channel, and resume communication in the new channel.

To facilitate channel surfing, both parties have to agree on the channel adaptation sequence, since A and B cannot negotiate with each other while they are within the jamming range. Additionally, we emphasize the importance of using orthogonal channels, since if A and B evade to a new channel that is not orthogonal to the original one, A or B will still be interfered with by X's signal. For many wireless networks, it is necessary to determine the number of orthogonal channels experimentally. For example, the specifications for the radio employed in Berkeley motes state that a channel separation of 150kHz is recommended in order to prevent cross-channel interference. As noted earlier, we have found through experiments that 800kHz is a safer value for channel separation in order to maintain *effective network-layer orthogonality*.

Another issue that naturally arises in employing such a channel changing strategy is whether or not one should continually change channels regardless of whether the adversary is blocking

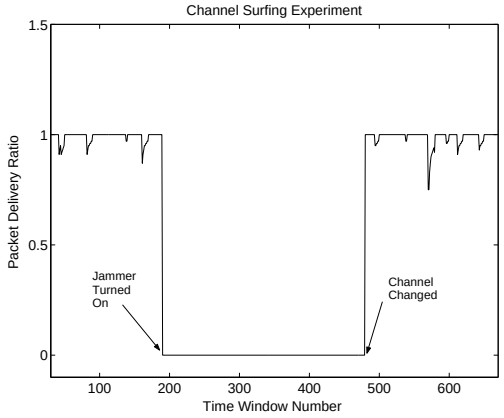


Figure 2.10: Packet delivery measurements from the Mote channel surfing prototype.

the current channel. Although physical layer frequency hopping employs a strategy of constantly changing the underlying frequency, there are reasons why this might not be desirable at the link-layer. In particular, although changing the frequency of the carrier wave is easy to accomplish in the case of frequency hopping spread spectrum, changing channels at the link-layer is more involved as it requires synchronization between both parties, which necessitates additional time cost.

Prototype: We built a proof-of-concept prototype system using two Berkeley motes A and B. The application running on these two motes involved A sending out a packet to B every 200 msecs. Each packet contained a sequence number starting at 1. We partitioned the time axis into windows, and B kept track of how many packets it received in each window (n_{recv}). It can also determine how many packets A has attempted to send in each window by looking at the sequence number of the last message it receives (n_{send}). In order to capture the quality-of-service of the application, we employed the *packet delivery ratio*, $r = n_{recv}/n_{send}$.

In the experiment, we used a waveform generator as the jammer X. As soon as the jammer is turned on, A cannot access the channel, and so no packets can be sent out. As a result, the packet delivery ratio becomes 0. In order for the application to survive the jamming, both A and B should incorporate a jamming detection and defense strategy. For the discussion here, our prototype detection algorithm works as follows. At the application level, each mote sets a jamming check timer (30 seconds). Each time the timer expires, it attempts to send out

a beacon by making a `SendMsg.Send()` call. The send call will return SUCCESS if the channel monitor component identifies an idle period so that the message send can start. (Later on, a notification will be sent to the application after the MAC-layer ACK is received from the receiver.) If the channel monitor component cannot sense the channel as idle after a long time (e.g., by using empirical threshold), the send call will return FAILURE. After it returns FAILURE continuously for several time, the mote can conclude that it is under a jamming attack. Following the detection of jamming, the mote will change its frequency to an orthogonal channel (e.g, from 916.7MHz to 917.6MHz). In order to avoid collision, A and B should not send beacons at the same time. The code is included below:

```
task void checkJamming(){
    sent = call SendMsg.send(TOS_BCAST_ADDR,
                             sizeof(uint16_t),
                             &beacon_packet);
    if(!sent){
        if(failures++ < thresh)
            post checkJamming();
        else post changeChan();
     }else{
        failures = 0;
    }
}
```

After both A and B change their frequencies, they can resume their application behavior, and the packet delivery ratio will go up again. We present measurements for the prototype channel surfing experiment in Figure 2.10. The channel surfing strategies works as expected.

2.3.2 Infrastructured Network

Now consider an infrastructured wireless network as depicted in Figure 2.9(b). Here, we have an access point AP_0, which has four wireless devices A, B, C, and D connected to it. There are two main scenarios for a radio interference/jamming attacks against the access point, corresponding to jammer X_0 and X_1. In the first scenario, jammer X_0 interferes with AP_0, A, B and C, but does not interfere with node D since it is outside of X_0's radio range. In the second scenario, jammer X_1 interferes with A and B, but not with AP_0 or any of the other nodes.

Algorithm: Infrastructured Channel Surfing

```
if DETECT_JAM(Self)==TRUE then
    Change_Channel()
else
    if AMI_AP == True then
        Calc_ChildrenLastCalledHome()
        Calc_NegligentChildren()
        RESPONSES = Probe_NegligentChildren()
        if Any(RESPONSES) == NULL then
            Broadcast_ChangeChannelCommand()
            Change_Channel()
        end
    else
        ListenForBeacons()
        if TimeToLastBeacon() > BIG then
            Change_Channel()
        end
    end
end
```

Algorithm 1: Channel surfing for wireless infrastructured networks. This algorithm runs on each network device.

The main difference between these two scenarios lies in the fact that one has the access point blocked by the jammer, while the other does not.

We need a strategy for changing channels whereby *all* nodes connected to the access point will change channels with the access point. We do not want to have scenarios where some devices are on the old channel while some are on the new channel. Algorithm 1 presents the sequence of events needed for a network device to determine whether to change channels.

Periodically, the algorithm checks to see whether the device has been blocked from communicating by a jammer. This can be done using the methods described in Section 4. If jamming has been detected, then the device will change channels. Otherwise, the device checks to see whether it is an access point. Access points will examine their list of children to see which devices have not communicated recently. Those devices whose last communication with the AP was greater than some threshold amount of time will be probed by the AP to ascertain whether they were remaining silent for a long time. If any device does not respond to their probe, the AP will conclude that the device has disappeared due to radio interference/jamming.

The rationale behind this is that, during normal operation of an infrastructured wireless network, if a network node wishes to leave the network it will perform a disassociation request, allowing the AP to free up any resources allocated to managing that network device. Further, when a network node moves to another access point, the device will perform reassociation with the new access point. The new access point will relay this information to the old access point,

```
Algorithm:Basic Ad Hoc Channel Surfing
if DETECT_JAM(Self)==TRUE then
    Change_Channel()
else
    RESPONSES = Random_ProbeNeighbors()
    if Any(RESPONSES == NULL) then
        STRANDED = Test_NextChannel()
        if Any(STRANDED)==TRUE then
            Broadcast_ChangeChannelCommand()
            Change_Channel()
        end
    end
end
```

Algorithm 2: Basic channel surfing for wireless ad hoc networks.

and acquire any data that might be buffered at the old access point. In both cases, the access point will know when a user legitimately leaves its domain.

If the AP concludes that the device disappeared due to jamming, then it will broadcast an emergency change channel packet that is signed by the AP's private key. This packet can be authenticated by each of the AP's children that are not blocked by the adversary. Following the issuance of the change channel command, the AP will change its channel and commence beaconing on the new channel in hopes to elicit associations from its children.

If the device is not an access point, then it will check to see if it has not heard its access point's beacons in a long time. It may even probe the access point. Either way, the child device will decide that it needs to catch up with its parent and change channels. In Algorithm 1, the default condition is to remain on the same channel. Performing an channel surfing procedure using Algorithm 1 allows the node A, B, C, and D in Figure 2.9(b) to survive all two jamming situations, regardless whether the AP is jammed or not. As a final comment, we note that in our overview discussion above, we have implicitly assumed that devices do not run out of battery or hardware failures. Thus, the issue of a client device disappearing will be solely due to interference.

2.3.3 Ad Hoc Network

When a jammer starts to interfere with an ad hoc network, he severs many of the links between network devices and can possibly cause network partitioning. Channel surfing can counteract such network faults by having the network or regions of the network switch to a new channel and re-establish network connectivity. In order to use channel surfing to address jamming for

Algorithm: Dual-Radio Ad Hoc Channel Surfing

```
if DETECT_JAM(Self)==TRUE then
    Change_Channel()
    InformNeighbors()
    EstablishNewLinks()
end
```

Algorithm 3: Dual-radio channel surfing for wireless ad hoc networks.

ad hoc networks, we assume that each network device keeps a neighbor list. However, since we are operating in ad hoc mode, we do not assume that if a device moves that it will inform its neighbors of its intent to relocate. Also, during unhindered network operation, we assume that no network partitions arise due to network mobility.

Algorithm 2 presents an outline of the channel surfing operations that run on each device. First, devices check to see if they have been blocked by a jammer. If so, they change channels and monitor the new channel to assist in reforming the ad hoc network on the new channel.

If a device has not been blocked by a jammer, then there is a chance that its neighbors have been blocked. Therefore, at random times a node will probe its neighbors to see that they are still nearby. There are several reasons that a neighbor node might not be present: it might have moved to a different part of the network, or it might have been blocked by a jammer. The device checks to see if a jammer is present by testing to see if any devices are stranded on the next channel. If the test returns positive, then the device returns to the original channel and broadcasts a signed change channel command to its neighbors, which is flooded through the rest of the network. It will then change channels and assist in reforming the ad hoc network on the new channel. Other devices will authenticate the command, and switch channels. If the test returned negative, then the device will assume that its absent neighbor has merely migrated to a different portion of the network and remove it from the neighbor list.

One unfortunate drawback of channel surfing for ad hoc networks is that it requires the use of flooding messages to promptly initiate a channel change across the entire ad hoc network. A simpler and more efficient alternative channel surfing strategy is possible if the network devices employ a dual-radio interface, that is they are capable of operating on two channels simultaneously.

Algorithm 3 describes the operations that run on each network device when a dual-radio interface is available. The default operation of the ad hoc network is for devices to employ

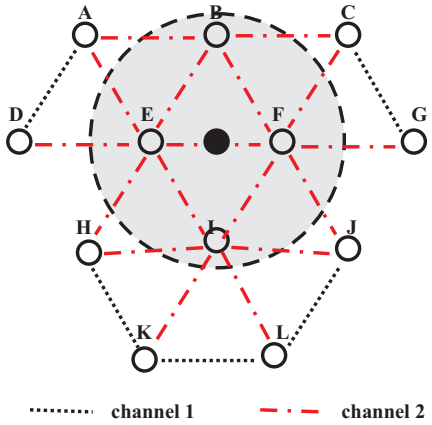


Figure 2.11: Channel surfing for an ad hoc network consisting of dual radio devices.

one radio channel for communication, yet monitor both channels. When a device detects that it has been blocked, it will switch to the next channel. Once on the new channel, the device will contact its neighbors via the new channel to inform them of its new channel policy, warn of possible radio interference/jamming, and establish new links in order to maintain network connectivity. In addition to the usual routing information, each network device must maintain an additional channel assignment field for each of its neighbors. The end result is that a network will consist of some links on the old channel and some links on the new channel, as depicted in Figure 2.11.

2.4 Spatial Retreats

The second escape strategy that we propose is *spatial retreats*. The rationale behind this strategy is that when mobile nodes are interfered with, they should simply move to a safe location. Spatial retreat is often a desirable defense strategy to employ in those wireless networks that involve mobile participants, such as users with cell phones or WLAN-enabled laptops. The key to the success of this strategy is to decide where the participants should move and how should they coordinate their movements. In this section, the discussion on spatial retreats primarily focuses on a simple interference scenario in which the jammer is stationary. This interference model might arise in cases where the jammer is unknowingly or unintentionally jamming the communication. More powerful adversarial models where the adversary is mobile and can stalk

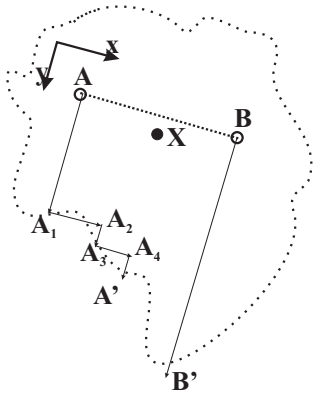


Figure 2.12: The spatial retreat strategy for a two-party communication scenario. The region depicted by the dotted line is the interference range of the adversary.

the communicating devices has been investigated and we refer the reader to [45].

2.4.1 Two-Party Radio Communication

Let us again start by examining the two-party communication scenario. We present an example jamming scenario in Figure 2.12, where the jammer X interferes with both A and B so that these two nodes cannot communicate with each other. In a spatial retreat, as soon as the communicating parties (i.e., A and B) detect the jamming scenario, they try to move away from the jammer. It is a daunting task, however, to decide on a retreat plan as both parties must agree on the direction of the retreat and how far to retreat. This task is complicated by the fact that A and B cannot communicate with each other while they are within the jammer's broadcast radio range. Further, even after they leave the adversary's radio range, they may not remain within each other's radio range due to the lack of synchronization between them and the irregularity of the interference region.

Considering the above factors, any functional retreat plan must satisfy the following two conditions: (1) it must ensure that both parties leave the adversary's interference range; and (2) it must ensure that the two parties stay within each other's radio range. In order to accomplish these two requirements, we propose a three-stage protocol:

1. *Establish Local coordinates:* We assume the two parties know each other's initial position prior to the introduction of the adversary. This assumption is reasonable since it

is becoming increasingly popular to incorporate positioning capabilities in mobile devices [32]. Using both parties' positions, we can decide on a local coordinate system (for example, we may define the x axis of our local coordinate system to be aligned with the segment $\overline{AB}$, as shown in Figure 2.12, and determine the y axis accordingly).

2. *Exit the Interference Region:* After the local coordinates are established, both parties move along the y-axis. While they move, they periodically check the interference level, e.g. the ambient signal strength or energy level emitted by the interferer. As soon as a node detects that it is out of the interference range, it stops moving. We would like to emphasize that, in practice, the two parties will stop asynchronously (as shown in Figure 2.12) because the radio range of the adversary is irregular in shape and the two parties cannot talk to each other before they move out of the range. In the example, A stops at location A_1, while B stops at B'.

3. *Move Into Radio Range:* There is a possibility, after exiting the adversary's radio range, that the two parties will be outside of each other's radio range, as shown in Figure 2.12. In this scenario, the two parties must move closer to each other so that they can resume their communication. If we let both parties move around, then they may not be able to find each other. Rather than giving both nodes the freedom to move in the third phase, we propose that one entity act as the Master, who will remain stationary, while the other entity acts as the Slave, who will move in search of the Master. In our figure, B is the Master and stays at B' while A moves to find B. Since every node knows the other node's initial location, A can move along the x-axis to approach B. One issue that comes up is that A may enter the interference range while searching for B. As soon as A detects that it is in the interference range, it must stop moving along the x-axis and return to moving along the y-axis to exit the interference range. While moving along the x-axis, A will not move beyond B's x-position, and if A's x-coordinate ever equals B's, A will move towards B directly.

This protocol achieves the two necessary conditions, and can easily be modified to handle scenarios where only one node is blocked by the interferer.

We studied the behavior of the proposed spatial retreat strategy by conducting a simulated radio communication scenario involving an adversary emitting a jamming signal in the

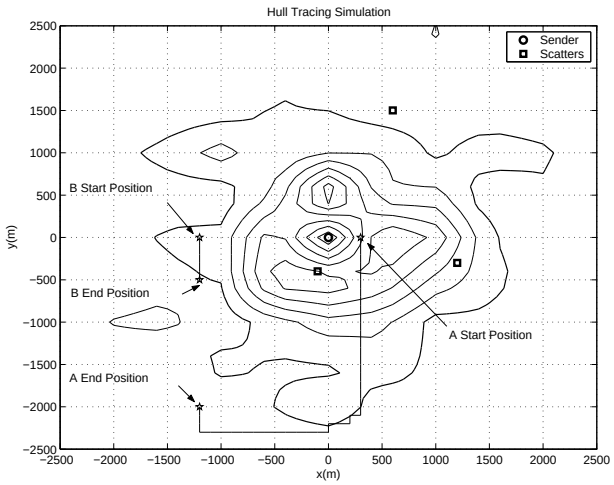


Figure 2.13: Simulated hull tracing scenario in which B is the Master and A is the Slave. Upon escaping the radio region of the adversary, A seeks to get within 1000 meters of B.

916.7 Mhz unlicensed band. The two entities A and B were initially located at $(300, 0)$ and $(-1200, 0)$ meters respectively. In order to capture a realistic non-isotropic radio pattern for the interferer, we placed three scatterers at $(600, 1500)$, $(1200, -300)$, and $(-100, -400)$ meters. The radio environment was simulated through ray tracing [23]. The scatterers were assumed to introduce random phase shifting in the transmitted signal. The hull tracing algorithm presented above was employed, with entity B acting as the Master while entity A acted as the Slave. Both A and B were assumed to know each other's initial position, and that they could measure the energy emitted by the interferer. We present the results of the simulation in Figure 2.13. In this figure, we have presented contours of equal energy for different (x, y) locations relative to the interferer. The paths taken by entities A and B are depicted. As both A and B escape the radio region of the adversary, A moves too far from B and cannot maintain radio connectivity. Therefore, A performs the *Move Into Radio Range* portion of the procedure.

2.4.2 Infrastructured Network

In the infrastructured scenario, there are several access points $AP_0, AP_1, \cdots, AP_N$ that are connected via a backbone. Wireless devices, R_j, connect to access points and perform communication between themselves (or with devices on the Internet) via routing through the APs.

As noted earlier, during jamming in the infrastructured network, the interferer can either

block the access point from the receivers, or block the receivers from communicating with the access point, or do both. A spatial retreat for an infrastructured wireless network must be a strategy that allows the user R_j to survive all three situations. The basic idea of spatial retreat in this context is that a mobile device will move to a new access point and reconnect to the network under its new access point. We note that it is not necessary for the access point to participate in a spatial retreat as access points are typically fixed infrastructure and not usually capable of mobility.

All three situations described above can be detected by an appropriate jamming detection strategy, as discussed in Section 4. In order to perform a spatial retreat for an infrastructured wireless network, we assume that each mobile device has an *Emergency Access Point List* assigned to it, and that the mobile device knows how to move in order to reach its Emergency Access Point. The Emergency Access Point can be assigned to each device by its current Access Point prior to the jamming/ interference.

When a device R_j detects that a jamming/interference occurs (either it cannot communicate, or it cannot receive beacons from its access point), it will begin to move towards its Emergency Access Point. While moving, it will occasionally pause and attempt to re-establish communication with its home access point. This is done in order to avoid any unnecessary handovers to other access points, and arises in scenarios where the adversary only blocks the user and not the access point. However, if R_j is not able to re-establish communication with its original AP, it will continue to move towards its Emergency Access Point. When R_j receives beacons from the new access point, it will initiate access point handoff. The purpose of access point handoff is to perform mutual authentication and establish authorization to use the new access point's services. There are many variations of authenticated handoff that can be employed, such as [21, 53].

We note that one problem that might result from spatial retreats in the infrastructured network is that all the mobile devices under an AP might move to the same Emergency AP. In order to prevent the resulting congestion at other APs, it is wise for the current AP to assign the Emergency Access Point lists in such a way as to divide the load across all of its neighboring APs.

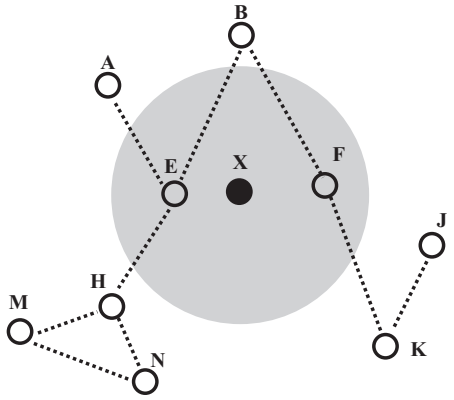


Figure 2.14: Scenarios for spatial retreat strategies in an ad hoc network setting. The adversary is marked by X.

2.4.3 Ad Hoc Network

It is much harder to design a spatial retreat strategy for ad hoc network scenarios because each node is not only involved in the communication it initiates but is also involved in forwarding packets. For ad hoc networks, it is critical to maintain network connectivity and if a node must leave its original position as a response to jamming attacks, it should move to a new location that minimizes degradation to network connectivity. For this preliminary study, we assume that only those nodes who are interfered with by the jamming attack need to escape, while other nodes should stay where they are. A globally optimized topology reformation strategy is beyond the scope of this paper [77].

Figure 2.14 illustrates a spatial retreat scenario in an ad hoc network setting, which attempts to minimize the network connectivity degradation. In Figure 2.14, node E originally connects to nodes A, B, and H, thus participating in flows $\overrightarrow{AEB}$, $\overrightarrow{AEH}$, and $\overrightarrow{HEB}$. After adversary X starts jamming the channel, E decides to move away. As shown in the figure, it is impossible for E to find a new location where it can avoid X but still maintain connection to A, B, and H. It has two choices: (1) move closer to A and B, or (2) move closer to A and H. (It cannot maintain $\overrightarrow{HEB}$ any more.) It compares these two options, and chooses the one which leads to a smaller loss in local network behavior by trying to maintain the local flow with the most value. Suppose the local flow $\overrightarrow{AEB}$ had a much higher traffic rate than the local flow $\overrightarrow{AEH}$. In this scenario, E decides to move to a new location between A and B in order to maintain the

high-valued flow $\overrightarrow{AEB}$. In the other example shown in the same figure, F connected B and K before jamming occurs. No matter where F moves to, it cannot connect both B and K. Network partitioning cannot be avoided in this case, and F should move to a location away from the interferer where it will be able serve as the endpoint for the most traffic.

We assumed that every node knows the location of its neighbors. This assumption can be realized by using equipment such as GPS. In addition, every node must keep track of the traffic rate of each stream it connects. Following a jamming/interference, each node will escape to a location where it can avoid the jammer, and continue to serve as much traffic as possible. Tracking the traffic rates is not an expensive operation, and can be accomplished by adding an additional column to the neighbor table for recording traffic measurements. This does not incur noticeable energy or memory overhead.

2.5 Summary and Conclusion

In this chapter, we have given an overview on the problem of jamming/radio interference. Our case studies for RF-interference show that jamming can be caused both by Non-MAC-compliant wireless devices and by MAC-compliant nodes. Non-MAC-compliant radio devices could jam network by either continuously sending packets ignoring the MAC-layer protocols, or just emitting jamming signals. The network nodes, which are compliant with the underlining MAC protocol but have lots of data packets to send, or have failed to choose a proper orthogonal channel, could interfere with other nodes in the network, as well.

We have discussed strategies to cope with non-MAC-compliant interferers. In particular, we have presented two different strategies that may be employed to mitigate the effects of interference. The rationale behind both strategies is that legitimate wireless users should avoid the interference as much as possible because it is very hard to combat the adversary. The first strategy involves changing the transmission frequency to a range where there is no interference from the adversary. The second strategy involves wireless users moving to a new location where there is no interference. We examined both strategies for three general classes of wireless networks: a generic two-party radio communication scenario, infrastructured wireless networks, and multi-hop ad hoc networks.

In the next chapter we examine the impact of jamming on wireless communication. We will then examine the issues of detecting interference in Chapter 4, and provide a more detailed analysis of channel surfing in ad hoc networks in Chapter 5.

Chapter 3

Jamming Attacks and Radio Interference

Jamming attacks/radio interference can severely interfere with the normal operation of wireless networks and, consequently, mechanisms are needed that can cope with jamming attacks/radio interference. The entity (either malicious or an unintentional wireless device) that launches such radio interference is referred to as the *jammer* or *interferer*. In the following three chapters, we examine radio interference from both sides of the issue: first, we study the problem of conducting radio interference on wireless networks in this chapter, and second we examine the critical issue of diagnosing the presence of jamming and resuming communication in the presence of jamming in Chapter 4 and Chapter 5, respectively.

The first stage towards defending a wireless network is to understand what is the impact that a jammer could have on wireless networks. Thus, in this chapter, we analyze the network performance degradation caused by radio interference using two methodologies, i.e. theoretical analysis, and real system implementation. In section 3.1, we first model the channel capacity under radio interference, and the geographical impact the radio interferers have on the networks according to the physical layer metrics signal to noise ratio. Then in section 3.2, we study the effectiveness of radio interference in real system implementation. In particular, we introduce four typical radio interference models, though by no means all-inclusive, which represent a broad range of radio interference strategies covering both intentional and unintentional radio interference in section 3.2.2. Finally, we implemented four jamming models and present the experimental results of their impact on communication in section 3.2.3.

3.1 Theoretical Analysis on the Effectiveness of Jamming

We start by examining the theoretical performance degradation caused by radio interference. In particular, we study the channel capacity in the presence of interference, and the geometric impact of the interferers based on Physical layer considerations.

There are many different strategies an interferer can use to jam wireless communications.

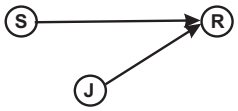


Figure 3.1: Two-party radio communication scenario.

For instance, the jammer either proactively jam the channel or reactively adjust its jamming strategy based on the network traffic J observes. We will describe the jamming strategies in Section 3.1.2. In this section, we focus our discussion on a proactive jammer, where the jammer continuously send out an interference signal.

3.1.1 Jamming Impact on Channel Capacity

The Shannon's capacity theorem tells us that the capacity of an additive white Gaussian noise channel is given by $C = Blog_2(1 + S/N)$, where B is the transmission bandwidth. S/N is the signal to noise ratio. There are several questions that arise regarding the effect of interference on the channel capacity.

Let's start by considering a simple network consisting of a source S, a receiver node R, and a jammer J that is interfering (either intentionally or accidentally) with legitimate wireless communications between S and R, as depicted in Figure 3.1. We assume the channel between node S and node R, and between jammer J and node R are both additive white Gaussian noise (AWGN) channels.

For simplicity, we shall abuse notation and let S be the signal sent by the sender S and J the one blasted by the jammer J. We assume that our signals are transmitted over a channel with bandwidth B. Without loss of generality, we assume the channel response for both the sender and the jammer is frequency non-selective (i.e. constant). This allows us to absorb the role of the $S \rightarrow R$ and $J \rightarrow R$ channel amplitude responses into the transmit power of S and J respectively, so that the received signal at node R is:

$$R = S + J + N, \tag{3.1}$$

where N is the Gaussian noise with variance σ^2, the signal S has power $P_s = 2B\sigma_s^2$ and the signal J has power $P_j = 2B\sigma_j^2$. The capacity of the communication channel between node S

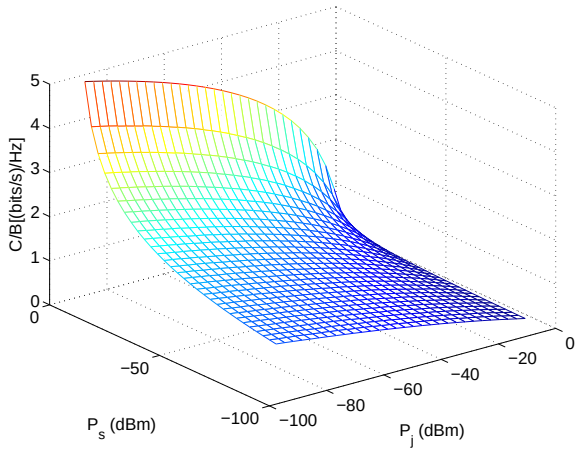


Figure 3.2: Normalized channel capacity as a function of P_s and P_j.

and node R is formally defined as

$$C = \max_{p(s)} I(S;R), \tag{3.2}$$

where $I(S;R)$ is the average mutual information that can be inferred about the transmitted signal S by observing the received signal R. The capacity of the channel, which is the maximum bit rate from $S \rightarrow R$, is the maximum value of $I(S;R)$ over all input symbol probability distributions. It is well-known that $I(S;R)$ is maximum when S is a Gaussian random variable [61], which is parameterized by the variance (power) term σ_s^2. Therefore, we shall assume that the sender's transmitted signal is a zero-mean Gaussian signal. Similarly, we also assume that the jammer is employs an optimal interference strategy, so that J's signal is a zero-mean Gaussian random variable with power σ_J^2. Straight-forward calculations give that the channel capacity as

$$\begin{aligned} C &= \max_{p(s)} I(S;R) \\ &= B\log_2(1+\frac{2\sigma_s^2}{2\sigma_j^2+2\sigma^2}). \end{aligned} \tag{3.3}$$

In the case where the noise power is much smaller than the jamming power, the channel capacity is approximately

$$C = B\log_2(1+\frac{P_s}{P_j}) \tag{3.4}$$

where P_s is the average transmission power of signal S, P_j is the average transmission power of jamming signal J. We plot the normalized capacity C/B versus the signal power P_s and the

jamming power P_j in Figure 3.2. For a given jamming power and bandwidth B, as the signal power P_s increases, the channel capacity increases. We note that the signal power P_s and the jamming power P_j are the one measured at the receiver end. Given that the sender retains its transmission power level the same, the closer the receiver is away from the sender, the larger P_s is, which suggests that moving the sender closer to the receiver while keeping the jammer where is was will increase the channel capacity. This naturally leads to the fact that the relative locations of the sender, the receiver, and the jammer affect the channel capacity. Thus, in the next subsection, we will study geographical issues associated with interference.

3.1.2 Non-isotropic Model for Jamming

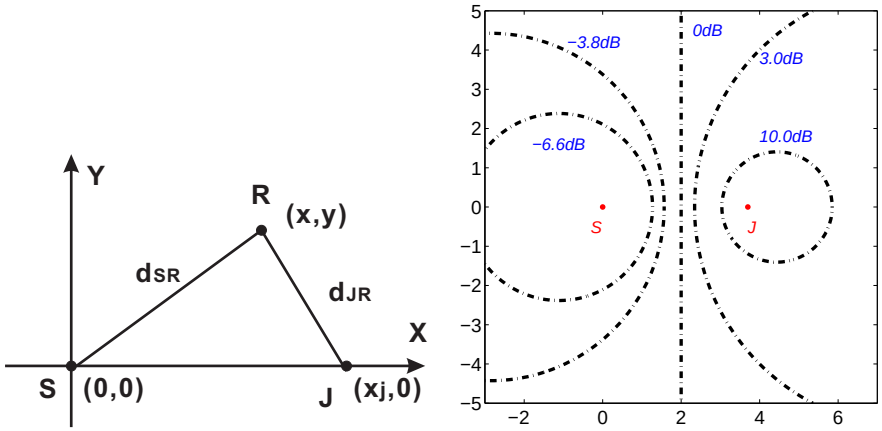
We now provide a geographical interpretation of the impact a jammer has on wireless communications. Typically, most recent papers on jamming and wireless networks have modeled the impact of the jammer as an isotropic effect. This has caused many authors to depict the effective jamming range as a circular region that is centered at the jammer's location. In reality, though, such an interpretation of jamming is overly simplistic and does not provide a complete depiction of the complex relationships between the transmission power of the source and the jammer, and the geometry of the deployment.

The circular jamming model does not capture the fact that the success reception of a packet is primarily determined by S/J at the receiver R. This ratio, S/J, depends on multiple factors, which include the transmission power of the source and jammer, as well as the distances between the receiver R, the source S, and the jammer J, i.e. d_{SR} and d_{JR}.

For a given pair of the source S and the jammer J, changing the location of the receiver R results in changing d_{SR} and d_{JR}, which in turn changes ratios S/J. To better understand the effectiveness of the jammer, we now derive the contours of constant S/J. Let $\gamma_{S/J}$ denote the signal-to-jamming ratio at each contour.

To begin, let us use the standard quadratic propagation loss model (i.e. a free space model). In this case, the received power is

$$P_R = \frac{P_T G_T G_R}{4\pi(d/d_0)^2} \tag{3.5}$$



(a) (b)

Figure 3.3: Coordinate system for constant S/J contours, (a) the positions of the sender S, the receiver R, and the jammer J; (b) constant S/J contour plot, where the distance between S and J is 4 units. The contour labels are the ratio of P_{ST}/P_{JT}, and the contour lines correspond to the edge where $\gamma_{S/J} = 0dB$.

where, P_T is the transmission power of the transmitter; G_T is the antenna gain of that transmitter in the direction of the receiver; G_R is the antenna gain of the receiver in the direction of the transmitter. d is the distance between the transmitter and the receiver, while d_0 is a reference distance (typically chosen so that $d_0 = 1$). Assume that the jammer uses the same type of device as the sender, e.g. both use omni-directional antennas. Then, this signal propagation model can be applied to the signal sent by the sender and the jammer as well, and the antenna gains G_T of the sender and the jammer are the same in all directions.

The signal-to-interference ratio at the receiver thus becomes

$$\gamma_{S/J} = \frac{P_{SR}}{P_{JR}} = \frac{P_{ST} d_{JR}^2}{P_{JT} d_{SR}^2}, \tag{3.6}$$

where P_{ST} is the transmission power of the sender S, P_{JT} is the transmission power of J. Using the coordinate system shown in Figure 3.3, the coordinate of the sender S $(0, 0)$, the jammer J is located at $(x_j, 0)$, and the receiver R is arbitrarily placed at (x, y). Noting that $d_{SR}^2 = (x_j - x)^2 + y^2$, and $d_{JR}^2 = x^2 + y^2$, we may substitute to get the contours of constant $\gamma_{S/J}$

$$(x - \frac{x_j}{1-\beta})^2 + y^2 = \frac{\beta x_j^2}{(1-\beta)^2}, \tag{3.7}$$

where $\beta = \frac{\gamma_{S/J}}{P_{ST}/P_{JT}}$. For a given P_{ST} and P_{JT}, the constant contours of $\gamma_{S/J}$ are circles centered at $(\frac{x_j}{1-\beta}, 0)$ with radius $\frac{\sqrt{\beta}x_j}{|1-\beta|}$.

We provide a depiction of equation (3.7) in Figure 3.3 (b). In this figure, we provide several contours corresponding to different transmit signal-to-jamming ratios, i.e. $P_{ST}/P_{JT} \in \{-6.6, -3.8, 0, 3.0, 10.0\}$ dB. Each of these contours map out loci of constant received signal-to-interference ratio corresponding $\gamma_{S/R} = 0dB$. We now discuss the interpretation of this figure. First, we note that the case $P_{ST}/P_{JT} = 0$dB splits the figure into two separate categories of curves: those centered near the source and those centered near the jammer. First, for those centered near the source, we may interpret these regions as areas where, if we were to place a receiver within one of these contours, it would be able to successfully decode transmissions from the source if the required signal-to-jammer ratio is $\gamma_{S/R} = 0dB$. Hence, a receiver located within the $P_{ST}/P_{JT} = -6.6$dB curve, would be able to decode packets (i.e. the receive signal-to-interference level would be better than 0dB) from a source that is transmitting at a level 6.6dB below the jammer. On the other hand, the circles centered near the jammer imply the contrary– a receiver within one of these circles would receive packets at a signal-to-interference level *worse* than $\gamma_{S/R} = 0dB$. For example, for those locations within the $P_{ST}/P_{JT} = 10$dB curve, even though the source transmits at a level 10dB higher than the jammer, the receiver is still unable to decode the transmission.

The contours depicted in Figure 3.3(b) shows that the effective jamming range cannot be simply modeled as a cycle centered at the jammer within which the receivers will be blocked. The effective jamming range depends on the network geographical topology. We note that, in a realistic deployment, the contours will not be so regular, due to the non-uniformity of the underlying multipath propagation environment. Thus, it is crucial to study the practical issues associated with the real system implementation.

3.2 System Study on Jamming/Interference Models and their Effectiveness

The first stage to defending a wireless network is to understand what types of radio interference are feasible, and how effective they are in a real system. Therefore, in this section, we introduce radio interference that may be implemented using wireless network devices. We first define the

characteristics of a jammer's behavior, and then enumerate metrics that can be used to measure the effectiveness of jamming. These metrics are closely related to the ability of a radio device to either send or receive packets. We then introduce four typical jammer attack models, though by no means all-inclusive, which represent a broad range of interference strategies that cover both intentional jamming attacks and unintentional radio interference, and will serve as the basis for our discussion throughout the remainder of the thesis. Throughout this thesis work, we will use the Berkeley MICA2 Mote platform for conducting our experiments with jammers. The observed characteristics of the jammers and the detection schemes presented later should hold for different wireless platforms, such as 802.11.

3.2.1 Jamming Characteristics and Metrics

Although several studies [52, 76, 77, 79] have targeted jamming-style attacks, the definition of this type of attack remains unclear. A common assumption is that a jammer continuously emits RF signals to fill a wireless channel, so that legitimate traffic will be completely blocked [77, 79]. We believe, however, that a broader range of behaviors can be adopted by a jammer. For example, a jammer may remain quiet when there is no activity on the channel, and start interference as soon as it detects a transmission. The common characteristic for all jamming attacks is that their communications are not compliant with MAC protocols. Therefore, *we define a jammer to be an entity who is trying to interfere with the physical transmission and reception of wireless communications either purposefully or accidentally.*

The outcome of a jammer is interference with legitimate wireless communications. A jammer can achieve this result by either preventing a real traffic source from sending out a packet, or by preventing the reception of legitimate packets. Let us assume that A and B denote two legitimate wireless participants, and let us denote X to be the jammer. A legitimate participant may be unable to send out packets for many reasons. To name just a couple, X can continuously emit a signal on the channel so that A will never sense the channel as idle, or X can keep sending out regular data packets and force A to receive junk packets all the time. On the other hand, however, even if A successfully sends out packets to B, it is possible for X to blast a radio transmission to corrupt the message that B receives. We thus define the following two metrics to measure the effectiveness of a jammer:

- **Packet Send Ratio (PSR):** The ratio of packets that are successfully sent out by a legitimate traffic source compared to the number of packets it intends to send out at the MAC layer. Suppose A has a packet to send. Many wireless networks employ some form of carrier-sensing multiple access control before transmission may be performed. For example, in the MAC protocol employed by Mica2, the channel must be sensed as being in an idle state for at least some random amount of time before A can send out a packet. Further, different MAC protocols have different definitions on an idle channel. Some simply compare the signal strength measured with a fixed threshold, while others may adapt the threshold based on the noise level on the channel. A radio interference attack may cause the channel to be sensed as busy, causing A's transmission to be delayed. If too many packets are buffered in the MAC layer, the newly arrived packets will be dropped. It is also possible that a packet stays in the MAC layer for too long, resulting in a timeout and packets being discarded. If A intends to send out n messages, but only m of them go through, the PSR is $\frac{m}{n}$. The PSR can be easily measured by a wireless device by keeping track of the number of packets it intends to send and the number of packets that are successfully sent out.

- **Packet Delivery Ratio (PDR):** The ratio of packets that are successfully delivered to a destination compared to the number of packets that have been sent out by the sender. Even after the packet is sent out by A, B may not be able to decode it correctly, due to the interference introduced by X. Such a scenario is an unsuccessful delivery. The PDR may be measured at the receiver B by calculating the ratio of the number of packets that pass the CRC check with respect to the number of packets (or preambles) received. PDR may also be calculated at the sender A by having B send back an acknowledge packet. In either case, if no packets are received, the PDR is defined to be 0.

3.2.2 Jamming Attack/Radio Interference Models

There are many different interference strategies that a jammer can perform in order to interfere with other wireless communications. As a consequence of their different interference intention

or attack philosophies, these various interference models will have different levels of effectiveness, and may also require different detection strategies. While it is impractical to cover all the possible interference models that might exist, in this study, we discuss a wide range of radio interference and somewhat malicious jamming attacks that have proven to be effective in disrupting wireless communication. Specifically, we have designed and built the following jammers:

- **Constant jammer:** The constant jammer continually emits a radio signal, and it can be either a malicious jamming attack where adversary purposely interfere with network communication, or unintentional radio interference where the interferer is always emitting RF signal. We have implemented a constant jammer using two types of devices. The first type of device we used is a waveform generator which continuously sends a radio signal. The second type of device we used is a normal wireless device. In this paper, we will focus on the second type, which we built on the MICA2 Mote platform. Our constant jammer continuously sends out random bits to the channel without following any MAC-layer etiquette. Specifically, the constant jammer does not wait for the channel to become idle before transmitting. If the underlying MAC protocol determines whether a channel is idle or not by comparing the signal strength measurement with a fixed threshold, which is usually lower than the signal strength generated by the constant jammer, a constant jammer can effectively prevent legitimate traffic sources from getting hold of channel and sending packets.

- **Deceptive jammer:** We use deceptive jammer to model a malicious jammer. Instead of sending out random bits, the deceptive jammer constantly injects regular packets to the channel without any gap between subsequent packet transmissions. As a result, a normal communicator will be deceived into believing there is a legitimate packet and will be duped to remain in the receive state. For example, in TinyOS, if a preamble is detected, a node remains in the receive mode, regardless of whether that node has a packet to send or not. Hence, even if a node has packets to send, it cannot switch to the send state because a constant stream of incoming packets will be detected. Further, we also observe that it is adequate for the jammer to only send a continuous stream of preamble bits (0xAA in

TinyOS) rather than entire packets.

- **Random jammer:** Instead of continuously sending out a radio signal, a random jammer alternates between sleeping and jamming. Specifically, after jamming for t_j units of time, it turns off its radio, and enters a "sleeping" mode. It will resume jamming after sleeping for t_s time. t_j and t_s can be either random or fixed values. During its jamming phase, it can either behave like a constant jammer or a deceptive jammer. Throughout this paper, our random jammer will operate as a constant jammer during jamming. The distinction between this model and the previous two models lies in the fact that this model tries to take energy conservation into consideration, which is especially important for those jammers that do not have unlimited power supply. By adjusting the distribution governing the values of t_j and t_s, we can achieve various levels of tradeoff between energy efficiency and jamming effectiveness. Random jammer can either represent a malicious jammer or unintentional interferer that interfere with network communication from time to time.

- **Reactive jammer:** Finally, we have a malicious jammer called reactive jammer. The three models discussed above are active jammers in the sense that they try to block the channel irrespective of the traffic pattern on the channel. Active jammers are usually effective because they keep the channel busy all the time. As we shall see in the following section, these methods are relatively easy to detect. An alternative approach to jamming wireless communication is to employ a reactive strategy. For the reactive jammer, we take the viewpoint that it is not necessary to jam the channel when nobody is communicating. Instead, the jammer stays quiet when the channel is idle, but starts transmitting a radio signal as soon as it senses activity on the channel. As a result, a reactive jammer targets the reception of a message. We would like to point out that a reactive jammer does not necessarily conserve energy because the jammer's radio must continuously be on in order to sense the channel. The primary advantage for a reactive jammer, however, is that it may be harder to detect.

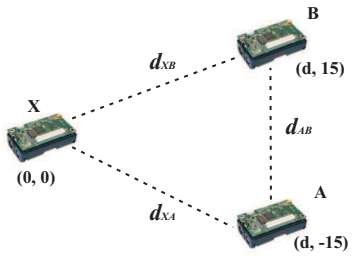


Figure 3.4: Placement of the Motes during jammer effectiveness experiments.

3.2.3 Experimental Results

We have implemented the above four jammer models using Berkeley Motes that employ a ChipCon CC1000 RF transceiver and use TinyOS as the operating system. We disabled channel sensing and back off operations to bypass the MAC protocol, so that the jammer can blast on the channel irrespective of other activities that are taking place. The level of interference a jammer causes is governed by several factors, such as the distance between the jammer and a normal wireless node, the relative transmission power of the jammer and normal nodes, and the MAC protocol employed by normal nodes. The closer a jammer is to a node, or the higher transmit power it employs, the greater the impact it will have on network operation. The MAC protocols employed by the network also play a role. Usually, MAC protocols decide the channel is idle if the measured signal strength value is lower than a threshold. Many MAC protocols, such as the one in TinyOS release 1.1.1, uses a fixed threshold value. Some MAC protocols, however, such as BMAC [58], adapt the threshold value based on the measured signal strength values, i.e. they choose the minimum signal strength among the most recent n readings when channel is idle as the current threshold value. Consequently, if a constant jammer transmits at a constant power, and both the jammer and the nodes are static, these adaptive MAC protocols will consider the channel as idle since they will regard the energy emitted by the jammer as ambient noise. In addition to these network configuration parameters, the impact of a jammer is also affected by jammer-specific parameters, such as the sleep interval for a random jammer. In order to understand the interactions of these parameters and quantify the impact of a jammer in different scenarios, we conducted a set of experiments involving three parties: A, B, and X, where A and B are normal wireless nodes with A being the sender, B the receiver, and X

Constant Jammer					
d_{XA} (inch)		BMAC		1.1.1 MAC	
		PSR (%)	PDR (%)	PSR (%)	PDR (%)
38.6		74.37	0.43	1.00	1.94
54.0		77.17	0.53	1.02	2.91
72.0		99.57	93.57	0.92	3.26
Deceptive Jammer					
d_{XA} (inch)		BMAC		1.1.1 MAC	
		PSR (%)	PDR (%)	PSR (%)	PDR (%)
38.6		0.00	0.00	0.00	0.00
54.0		0.00	0.00	0.00	0.00
72.0		0.00	0.00	0.00	0.00
Random Jammer					
d_{XA} (inch)		BMAC		1.1.1 MAC	
		PSR (%)	PDR (%)	PSR (%)	PDR (%)
t_j = U[0,31] t_s = U[0,31]	38.6	79.45	0.26	70.19	16.77
	44.0	80.15	17.48	70.30	21.95
	54.0	80.43	99.00	76.98	99.75
t_j = U[0,31] t_s = U[1,8]	38.6	60.47	0.06	56.49	0.00
	44.0	60.72	47.41	56.00	0.41
	54.0	61.77	96.75	100.0	99.64
Reactive Jammer					
d_{XA} (inch)		BMAC		1.1.1 MAC	
		PSR (%)	PDR (%)	PSR (%)	PDR (%)
m = 7bytes	38.6	99.00	0.00	100.0	0.00
	54.0	100.0	99.24	100.0	99.87
	72.0	100.0	99.35	100.0	99.97
m = 33bytes	38.6	99.00	0.00	100.0	0.00
	44.0	99.00	58.05	100.0	87.26
	54.0	99.25	98.00	100.0	99.53

Table 3.1: The resulting PSR and PDR for different jammer models under various scenarios.

a jammer using one of our four models. The transmission power levels employed by A, B, X are all $-4dBm$. These three nodes are carefully placed so that X has the same impact on both A and B. In particular, we set d_{XA}, the distance between X and A, equal to d_{XB}, the distance between X and B, and we fixed the distance between the sender A and the receiver B at $d_{AB} = 30$ inches, as depicted in Fig. 3.4.

The resulting PSR and PDR for each jammer model are summarized in Table 3.1. As the Table 3.1 shows, if A employs 1.1.1 MAC, a constant jammer that is reasonably close to A can completely block A, from sending out packets, resulting in a very low PSR. However, if A employs BMAC, which adapts the threshold based on the surrounding signal strength, A can still manage to send out a large portion of the packets, i.e, with PSR being 74.37% even when X is only 38.6 inches away from A. The reason why A cannot send out all of the packets is that the signal strength produced by X varies with time. The corresponding PDR in both cases, however, is poor because most of the packets are corrupted by the constant jammer, especially when the constant jammer is close to the sender.

However, the same trend cannot be observed for a deceptive jammer. Since a deceptive jammer continuously sends out packets with valid preamble, both A and B are forced to constantly stay in the reception mode no matter which MAC protocol they use. Hence, A and B cannot send out any packets at all and the PSR are 0% all the time. PDR in this case is defined as 0.

For the random jammer, in addition to studying the impact of network configuration parameters, such as the distance between the jammer and the nodes, and the MAC protocol on the effectiveness of the jammer, we also look at jammer-specific parameters, such as the on-off periods. Specifically, we studied two random jammers. For the first random jammer, the duration of the jamming period t_j is a uniform random number between 0 and 31 spibus interrupts in TinyOS [25], denoted by t_j = U[0,31], and the duration of the sleeping period t_s is a uniform random number between 0 and 31 as well, denoted by t_s = U[0,31]. For the second random jammer, t_j = U[0,31], and t_s = U[1,8]. On average, the second jammer sleeps less, and switches to the jamming mode more often. Thus, the PSR measured in the second jammer scenario is less than the PSR in the first jammer scenario. Additionally, since the random jammer alternates between jamming and sleeping, BMAC, which always chooses the minimum signal strength value among the recent readings, cannot increase the threshold quickly enough to consider the channel idle. Thus, BMAC considers the channel as busy when the random jammer is jamming, resulting in a lower PSR.

A reactive jammer starts interference as soon as it hears a transmission on the channel. Consequently, the effectiveness of a reactive jammer is also dependent on size of legitimate network packets as well as the size of packet the jammer emits. In Table 3.1, we explore the behavior of the reactive jammer for network packets of size $m = 7$ and $m = 33$ bytes, where the jammer emitted a 20 byte jamming packet. First, we observe that in all cases the sender is able to reliably send out its packets. Ideally, if m is short, one would infer that there may not be enough time for a reactive jammer to corrupt a network packet in transmission. However, as we see in Table 3.1, for different network packet sizes, although there is a difference in the resulting PDR, the difference is in fact negligible. Hence, even for short packets of a few bytes in length, a jammer employing the reactive strategy is able to effectively disrupt network communication.

3.3 Summary and Conclusion

The shared nature of the wireless medium will allow adversaries to pose non-cryptographic security threats by conducting radio interference attacks. Therefore, understanding the nature of jamming attacks and radio interference is critical to assuring the operation of wireless networks. Thus, in this chapter, we have analyzed the network performance degradation caused by radio interference from two aspects, i.e. theoretical analysis, and real system implementation.

First, we have studied channel capacity in radio interference scenarios theoretically. The channel capacity is the average mutual information that can be inferred from the received signal. Our theoretical analysis shows that the channel capacity is given by $C = B\log_2(1 + \frac{P_s}{P_j})$, where B is the transmission bandwidth, P_s is the average transmission power of signal S, and P_j is the average transmission power of jamming signal J. Starting from the fact that the channel capacity is a function of $\frac{P_s}{P_j}$, we have derived the contour of $\frac{P_s}{P_j}$ which represents the geographical impact that a jammer has on wireless communications. Our study shows that the effective jamming range cannot be simply modeled as a circle centered at the jammer. Instead, the effective jamming range depends on the geographical location of the source, the jammer and the receiver.

Theoretical study gives an insight on the impact that an interferer can have on the network. It, however, does not capture many practical issues. In the second part of this chapter, we have studied the effectiveness of radio interference in a real system implementation. In particular, we introduce four typical radio interference models, though by no means all-inclusive, which represent a broad range of radio interference strategies, and cover both intentional and unintentional radio interference. We implemented four jamming models and the experiment results show that they are effective in disturbing the network communications.

Chapter 4

Detecting Jamming Attacks and Radio Interference

Detecting jamming attacks is important because it is the first step towards building a secure and dependable wireless network. *Detecting radio interference attacks is challenging as it involves discriminating between legitimate and adversarial causes of poor connectivity.* Specifically, we need to differentiate a jamming scenario from various network conditions: congestions that occur when the aggregated traffic load exceeds the network capacity so that the packet send ratio and delivery ratio are affected; the interrupt of the communication due to failures at the sender side; and other similar condition.

In this chapter, we address the problem of detecting jamming attacks and radio interference. In particular, we focused on detecting the four jamming models presented in Chapter 3, which cover a broad range of malicious jamming attacks and unintentional radio interference. There are several measurements that naturally lend themselves to detecting jamming, such as signal strength, carrier sensing time, and packet delivery ratio. In Section 4.1, we explore these measurements in detail and present scenarios where they may not be effective in detecting a jamming attack, and in fact could cause false detections. For each of these measurements, we develop statistics upon which to make decisions. Since statistics built upon individual measurements may lead to false conclusions, in Section 4.2 we develop two improved detection strategies. These two detection strategies are both built upon the fundamental assumption that communicating parties should have some basis for knowing what their characteristics should be if they are not jammed, and consequently can use this as a basis for differentiating jammed scenarios from mere poor link conditions.

4.1 Basic Statistics for Detecting Jamming Attacks and Radio Interference

4.1.1 Signal Strength

One seemingly natural measurement that can be employed to detect jamming is signal strength, or ambient energy. The rationale behind using this measurement is that the signal strength

distribution may be affected by the presence of a jammer. In practice, since most commodity radio devices do not provide signal strength or noise level measurements that are calibrated (even across devices from the same manufacturer), it is necessary for each device to employ its own empirically gathered statistics in order to make its decisions. Each device should sample the noise levels many times during a given time interval. By gathering enough noise level measurements during a time period prior to jamming, network devices can build a statistical model describing normal energy levels in the network.

We now explore two basic strategies that employ signal strength measurements for detecting a jamming attack and radio interference. The first approach uses either the average signal value or the total signal energy over a window of N signal strength measurements. This is a simple approach that extracts a single statistic for basing a hypothesis test upon. Since a single statistic loses most of the shape characteristics of the time series, a second strategy would seek to capture the shape of the time series by representing its spectral behavior. The second strategy that we discuss uses N samples to extract spectral characteristics of the signal strength for the basis of discrimination. In the discussion below, we assume that we have measured the channel's received energy levels $s(t)$ at different times and collected N of these samples to form a window of samples $\{s(k), s(k-1), \cdots, s(k-N+1)\}$.

Basic Average and Energy Detection

We can extract two basic statistics from signal strength readings, namely, the average signal strength and the energy for detection. In both cases, the statistical hypothesis testing problem is binary and essentially involves deciding between signal absent and signal present hypotheses.

The use of the signal average arises naturally when the jammer emits a constant amplitude signal. In this case, the detection statistic is $T(k) = (\sum_{j=k-N+1}^{k} s(j))/N$. The use of the signal energy arises when the jammer emits a powerful noise-like signal, such as a white Gaussian process. Here, the detection statistic is $T(k) = (\sum_{j=k-N+1}^{k} s(j)^2)/N$. In either case, the detection decision is made by comparing $T(k)$ to a threshold γ that is suitably chosen by considering tradeoffs between probability of detection and false alarm, such as through application of the Neyman-Pearson theorem [33, 59].

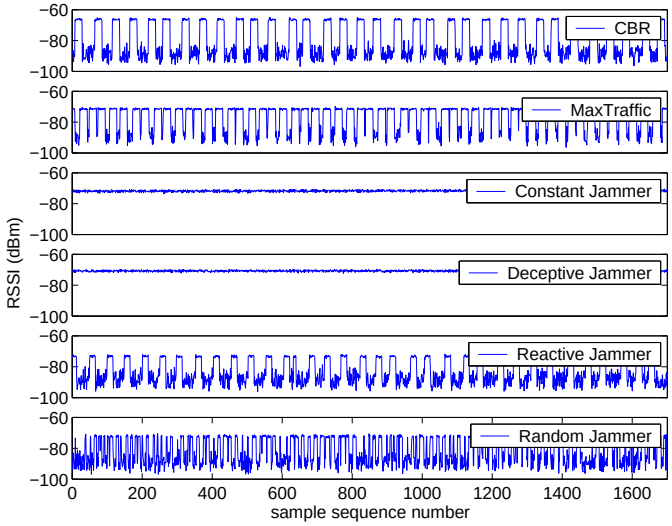


Figure 4.1: RSSI readings as a function of time in different scenarios. RSSI values were sampled every 1msec.

Signal Strength Spectral Discrimination

The average signal strength or the signal energy over a window of N samples does not reflect the fact that there may be many different received signal sample paths that could have led to the same mean or energy value. For example, a signal that has half of its ADC values as 50 and half as 150 would be considered the same as a signal whose samples are all 100 if we use the average signal strength as our decision statistic.

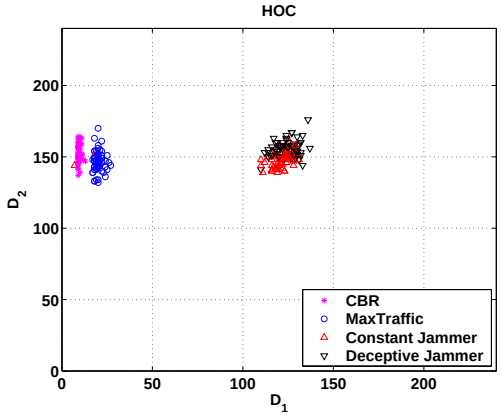
In order to have more robustness to false decisions and enhance the ability to classify scenarios, it is natural to use spectral discrimination techniques to classify the signal. One possible spectral discrimination mechanism is to employ higher order crossings (HOC). We refer the reader to the treatise on HOC [34] for explicit definition of HOC statistics. We have chosen to study higher order crossings since the calculation of these statistics only involves differences between samples, and is thus simple and practical to implement on resource-constrained wireless devices, such as sensor nodes. More complicated spectral techniques that involve the estimation of power spectral densities are possible and yield comparable performance but require more computational complexity.

Effectiveness Analysis:

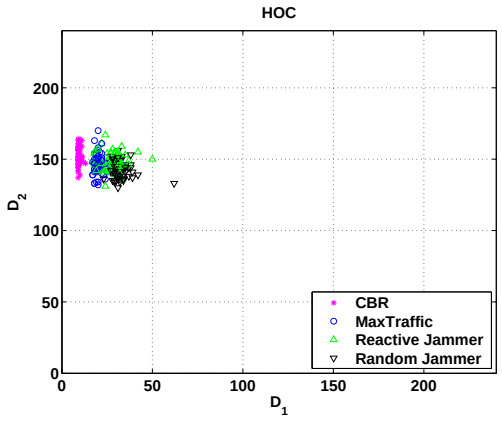
In order to understand the effect that a jammer would have on the received signal strength, we performed six experiments. In the first two experiments, we have two Motes, a sender A and a receiver B, which are 30 inches apart from each other. In the first case, A transmits 20 packets per second, corresponding to a traffic rate of 5.28kbps, which we refer to as a CBR source. In the second case, A transmits at its maximum rate; as soon as the send function returns to the application level asynchronously, either because the packet is successfully sent or because the packet is dropped (the packet pumping rate is larger than the radio throughput), it posts the next send function. Such a sender is referred to as a MaxTraffic source, and corresponds to a raw traffic rate of 6.46kbps. In the following four experiments, in addition to A and B, we introduced the jammer X, which was placed 54 inches away from B, with X employing our four jammer models. When X behaves as a random jammer, it uses the following parameters: t_j = U[0,31] and t_s = U[0,31]. In these four jamming scenarios, A is a CBR source. In each of these six experiments, the receiver B obtains the RSSI values by posting the `RSSIADC.getData()` function on the port `TOS_ADC_CC_RSSI_PORT` every millisecond. The reported RSSI values in Fig. 4.1, in dBm, are converted from the raw values following the analog-to-digital conversion of the received voltage levels [1]. We present time series data for each of the six scenarios in Fig. 4.1. From these results, we observed that the average values for the constant jammer and the MaxTraffic source scenario, are roughly the same. Further, the constant jammer and deceptive jammer have roughly the same average values, with the slight difference in the plot resulting from experimental setup. Additionally, the signal strength average from a normal CBR source does not differ much from that measured for the reactive jammer scenario. Similar statements can be made for using the signal energy. These results suggest the following important observation: we may not be able to use simple statistics, such as average signal strength or energy, to discriminate jamming scenarios from normal traffic scenarios because it is not straightforward to devise a threshold that can separate these two scenarios.

There is a practical issue that arises from the locations the nodes and jammers relative to each other. Nodes that are very close to each other will naturally lead to high signal strength measurements, while nodes separated by more distance will yield lower signal strength measurements.

From the time series in Fig. 4.1, we observe that there are some differences in the shapes



(a)



(b)

Figure 4.2: Plot of the first two higher order crossings, D_1 vs. D_2, for different jammer and communication scenarios.

underlying the time series for these scenarios. For example, the measured signal strength for the constant jammer and the deceptive jammer exhibit a much lower variation (the time series curve is almost flat) compared to the signal strengths for MaxTraffic source.

We next examined the issue of whether spectral discrimination techniques would be able to distinguish between normal and jammed scenarios. We calculated the first two higher order crossings for the time series, D_1 and D_2, using a window of 240 samples. We plot D_1 versus D_2 in Fig. 4.2. From the Fig. 4.2 (a), we observe that the points gather in two clusters, one cluster corresponding to the constant and deceptive jammers, while the other cluster corresponding to normal CBR and MaxTraffic sources. Hence, using HOC, we can distinguish normal traffic scenarios from the constant and deceptive jammer. However, examining Fig. 4.2 (b) we see

that we cannot distinguish the reactive or random jammer from normal traffic scenarios. The reason for this is that a reactive jammer or random jammer causes the channel state to alternate between busy and idle in much the same way as normal traffic behaves. In particular, because the reactive jammer does not change the underlying busy and idle periods for a normal traffic scenario, it is particularly difficult to distinguish between signal readings for a reactive jammer and signal readings from the underlying traffic.

Hence, based on these observations, we conclude that employing HOC (or even other spectral methods), will work for some jammer scenarios, but are not powerful enough to detect all jammer scenarios.

4.1.2 Carrier Sensing Time

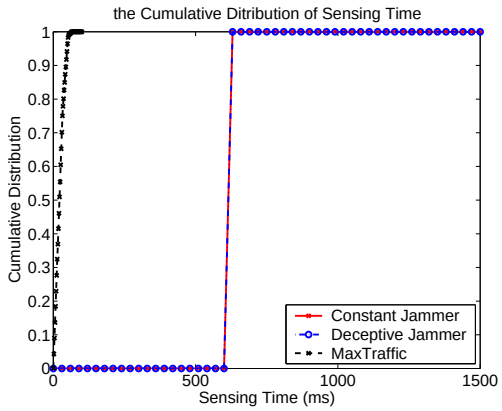
As discussed in Chapter 3.2, a jammer can prevent a legitimate source from sending out packets because the channel might appear constantly busy to the source. In this case, it is very natural for one to keep track of the amount of time it spends waiting for the channel to become idle, i.e. the carrier sensing time, and compare it with the sensing time during normal traffic operations to determine whether it is jammed. We would like to emphasize that this is only true if the legitimate wireless node's MAC protocol employs a fixed signal strength threshold to determine whether the channel is busy or idle. For protocols that employ an adaptive threshold, such as BMAC, after the threshold has adapted to the ambient energy of the jammer, the carrier sensing time will be small even when a jammer is blasting on the channel. Consequently, in the rest of this section, we only focus on MAC protocols that employ a fixed threshold, such as the MAC in TinyOS 1.1.1.

In most forms of wireless medium access control, there are rules governing who can transmit at which time. One popular class of medium access control protocols for wireless devices are those based on carrier sense multiple access (CSMA). CSMA is employed in MICA2 Motes as well as in both infrastructure and infrastructureless (ad hoc) 802.11 networks. The MAC-layer protocol for 802.11 additionally involves an RTS/CTS handshake. During normal operation of CSMA, when A (the sender) tries to transmit a packet, it will continually sense the channel until it detects the channel is idle, after which it will wait an extra amount of time (known as the propagation delay) in order to guarantee the channel is clear. Then, if RTS/CTS

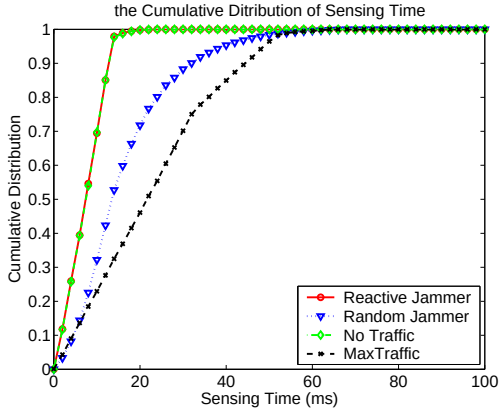
is used it will send the RTS packet, or otherwise will send the data packet. Suppose we assume that the jammer X continuously emits radio signal on a channel and that A attempts to transmit a packet. Then, since the channel is occupied by X, A will either time-out the channel sensing operation (if a time-out mechanism is available in the MAC protocol) or be stuck in the channel sensing mode.

Unfortunately, a large carrier sensing time could have occurred in non-jammed scenarios as well, such as congestion. It is therefore important to have some mechanism to distinguish between normal and abnormal failures to access the channel. In order to do so, a thresholding mechanism based on the sensing time can be used to identify jamming: Each time A wishes to transmit, it will monitor the time spent sensing the channel, and if that time is above a threshold (or if it is consistently above the threshold), it will declare that a jamming is occurring. The threshold may be determined theoretically based on a simple channel occupancy model, or empirically. The problem with theoretically calculating the threshold is that it is extremely difficult to build a complete mathematical model that captures a realistic MAC protocol. A well-known $M/M/1/1$ queuing model may be used to describe the MAC protocol [35, 36, 79], but doesn't capture the notion of collisions, or retransmissions. Therefore, we focus on the second approach to determining the threshold, which involves each network device collecting statistics regarding the amount of time D that a device must wait before it can start transmission during normal, or even somewhat congested, network conditions. With a distribution $f_D(d)$ describing carrier sensing times during acceptable network conditions, we may classify any new measured sensing time as either normal or anomalous by employing significance testing [59]. In this case, our null hypothesis is that the measured delay D corresponds to the distribution $f_D(d)$. If we reject the null hypothesis, then we conclude the network is experiencing a jamming attack. Since it is undesirable to falsely conclude the presence of jamming when the network is merely experiencing a glitch, we need to use a conservative threshold to reduce the probability of a false positive.

Effectiveness Analysis: In order to quantify the validity of detecting jamming at the MAC-layer using carrier sensing time in a real wireless network, we performed an experiment using two Motes, X and A. Here, Mote A corresponds to a network node trying to send out a



(a)



(b)

Figure 4.3: The cumulative distribution for the carrier sensing times measured using MICA2 Motes.

33-byte packet every 100msecs, and which measures the sensing time while doing so. Mote A employed the MAC protocol from TinyOS release 1.1.1, which used a fixed threshold for determining idleness. Mote X cycles through the four different types of jammers, as well as the MaxTraffic source. Additionally, we measured the sensing time when there is no background traffic, i.e. X does not send any traffic.

Fig. 4.3 depicts the cumulative distribution of the sensing time for the six different scenarios. Fig. 4.3 (a) shows that the cumulative distribution of the constant jammer and the deceptive jammer jumps at the point where the sensing time equals to 640msces. This is caused by a time-out we added to the TinyOS. In our experiment, if the device does not start to send the packet

within 640msecs after the packet was passed to the MAC-layer from application layer, a time-out will occur, the packet will be discarded, and its sensing time will be counted as 640msecs.

The drawback of carrier sensing time is that it exhibits significant missed detections in the presence of other types of jammers. As Fig. 4.3 (b) shows, most of the sensing time in other jammer scenarios is smaller than the sensing time in a congested scenario. The reactive jammer will exhibit normal carrier sensing times because the jammer will not attempt to jam until another node has successfully started transmission. As a result, the transmitting node A will observe normal carrier sensing times. In particular, in our experiment the reactive jammer produces sensing time cumulative distributions that overlap completely with the case of no background traffic.

We note that, if the MAC protocol employs an adaptive threshold for determining channel idleness, instead of the fixed threshold in our experiment, then the cumulative distribution of the sensing time for the constant jammer would have shifted to the left, while there would have been no difference for the deceptive jammer since node A would still have been locked in a received state. The reactive jammer would have exhibited the same characteristics. Similar to the constant jammer, the random jammer also shifts the cumulative distribution to the left. We verified these observations through identical experiments to the ones described above where we used BMAC instead of the MAC protocol from TinyOS release 1.1.1.

In summary, both signal strength and carrier sensing time, under certain circumstances, can only detect the constant jammer and deceptive jammer. Neither of these two statistics is effective in detecting the random jammer or the reactive jammer.

4.1.3 Packet Delivery Ratio

A jammer may not only prevent a wireless node from sending out packets, but may also corrupt a packet in transmission. Consequently, we next evaluate the feasibility of using packet delivery ratio (PDR) as the means of detecting the presence of jamming. The packet delivery ratio can be measured in the following two ways: either by the sender, or by the receiver. At the sender side, the PDR can be calculated by keeping track of how many acknowledgements it receives from the receiver. At the receiver side, the PDR can be calculated using the ratio of the number of packets that pass the CRC check with respect to the number of packets (or preambles) received.

Unlike signal strength and carrier sensing time, PDR must be measured during a specified window of time where a baseline amount of traffic is expected. If no packet is received over that time window, then the PDR within that window is zero.

Since a jamming attack will degrade the channel quality surrounding a node, the detection of a radio interference attack essentially boils down to determining whether the communication node can send or receive packets in the way it should have had the jammer not been present. More formally, let us use π_0 to denote the PDR between a sender and a receiver, who are within radio range of each other, assuming that the network only contains these two nodes and that they are static. As shown in Table 3.1, any one of the four jammers, if placed within a reasonable distance from the receiver, can cause the corresponding PDR to become close to 0. In the cases shown in Table 3.1, π_0 is 100%. From these results, we can conclude that a jammer can cause the PDR to drop significantly. We would like to point out that a non-aggressive jammer, which only marginally affects the PDR, does not cause noticeable damage to the network quality and does not need to be detected or defended against.

Next, we need to investigate how much PDR degradation can be caused by non-jamming, normal network dynamics, such as congestion, failures at the sender side, etc. In order to study the impact of congestion on PDR, we introduced 3 MaxTraffic sources, resulting in a raw offered traffic rate of 19.38kbps[1], to model a rather highly congested scenario. Even under such a congestion level, the PDR measured by the receiver is still around 78%. As a result, a simple thresholding mechanism based on the PDR value can be used to differentiate a jamming attack, regardless of the jamming model, from a congested network condition.

Though PDR is quite effective in discriminating jamming from congestion, it is not as effective for other network dynamics, such as a sender battery failure, or the sender moving out of the receiver's communication range, because these dynamics can result in sudden PDR drop in much the same way as a jammer does. Specifically, if the sender's battery drains out, it stops sending packets, and the corresponding PDR is 0%.

Consequently, compared to signal strength and carrier sensing time, PDR is a powerful

[1]At 100% duty cycle, the MICA2 radio's maximum bandwidth capacity is 12.364kbps, though the effective maximum throughput is typically much less than that.

statistic in that it can be used to differentiate a jamming attack from a congested network scenario, for different jammer models. However, it still cannot differentiate the jamming attack from other network dynamics that can disrupt the communication between the sender and the receiver.

4.2 Jamming Detection with Consistency Checks

In the previous section we saw that no single measurement is capable of detecting all kinds of jamming attacks. Since the purpose of a jammer is to influence the channel quality between a node and its neighbors, it is not reasonable, or needed, to try to detect a jammer if that jammer does not effectively interfere with the receipt/send of packets at a node. While a node losing its sending ability is a clear sign that it is being jammed, a weak reception capability (i.e. a low PDR) can be caused by several factors besides jamming, such as a low link quality due to the relatively large distance between the sender and the receiver.

We observed in the previous section that PDR is a powerful measurement that is capable of discriminating between jammed and congested scenarios, yet is unable to identify whether an observed low PDR is due to natural causes of poor link quality. In order to compensate for this drawback, and enhance the likelihood of detection, we will examine two strategies that build upon PDR to achieve enhanced jammer detection. We augment the use of PDR by applying signal strength measurements to conduct consistency checking to determine whether low PDRs are due to natural causes or due to radio interference. Later, in Section 4.2.2, we discuss a complementary technique that uses location information to augment PDR measurements for jamming detection.

Throughout this section, we assume that a node is only responsible for detecting whether it is jammed, and is not responsible for detecting the jammed condition of its neighbors. This follows from the fact that a wireless node is the best source of information regarding its local radio environment and is a less reliable predictor of the radio condition at distant locations. We assume that each node maintains a neighbor list, obtained from the routing layer, which will assist in making more reliable detection decisions. Additionally, we assume that the deployment of the network is sufficiently dense to guarantee that each node has several neighbors.

Algorithm:PDRSS_Detect_Jam
$\{PDR(N) : N \in Neighbor\_list\}$ = Measure_PDR()
$MaxPDR = \max\{PDR(N) : N \in Neighbor\_list\}$
if $MaxPDR < PDRThresh$ **then**
  SS = Sample_Signal_Strength()
  $CCheck$ = SS_ConsistencyCheck($MaxPDR$, SS)
  if $CCheck ==$ *False* **then**
    post NodeIsJammed()
  end
end

Algorithm 4: Jamming detection algorithm that checks the consistency of PDR measurements with observed signal strength readings.

All legitimate nodes in the network will participate in the detection protocol by transmitting a baseline amount of traffic, e.g. by sending heartbeat beacons. This allows each node to reliably estimate PDR over a window of time, and conclude that the PDR is 0 if no packets are observed during that time period.

4.2.1 Signal Strength Consistency Checks

The packet delivery ratio serves as our starting point for building the enhanced detector. Rather than rely on a single PDR measurement to make a decision, we employ measurements of the PDR between a node and each of its neighbors. In order to combat false detections due to legitimate causes of link degradation, we use the signal strength as a consistency check. Specifically, we check to see whether a low PDR value is consistent with the signal strength that is measured. In a normal scenario, where there is no interference or software faults, a high signal strength corresponds to a high PDR. However, if the signal strength is low, which means the strength of the wireless signal is comparable to that of the ambient background noise, the PDR will be also low. On the other hand, a low PDR does not necessarily imply a low signal strength. It is the relationship between signal strength and PDR that allows us to differentiate between the following two cases, which were not possible to separate using just the packet delivery ratio. First, from the point of view of a specific wireless node, it may be that all of its neighbors have died (perhaps from consuming battery resources or device faults) or it may be that all of a node's neighbors have moved beyond a reliable radio range. A second case would be the case that the wireless node is jammed. The key observation here is that in the first case, the signal strength is low, which is consistent with a low PDR measurement. While in the jammed case, the signal strength should be high, which contradicts the fact that the PDR is low. Table 4.1 summarizes

Observed PDR	Observed signal strength	Typical scenarios
PDR = 0 (no preamble is received)	low signal strength	non-jammed: neighbor failure, neighbor absence, neighbors being blocked, etc.
PDR = 0 (no preamble is received)	high signal strength	node jammed
PDR low (packets are corrupted)	low signal strength	non-jammed: neighbor being faraway
PDR low (packets are corrupted)	high signal strength	node jammed

Table 4.1: A combination of PDR and signal strength improves jamming detection accuracy.

typical network scenarios that can cause low PDR values and how the signal strength measurements can help further isolate the cause of the low PDR values.

Based on these observations we propose the detection protocol shown in Algorithm 4. In the `PDRSS_Detect_Jam` algorithm, a wireless node will declare that it is not jammed if at least one of its neighbors has a high PDR value. However, if the PDRs of all the neighbors are low, then the node may or may not be jammed and we need to further differentiate the possibilities by measuring the ambient signal strength. Rather than continually sample the ambient signal levels, which may use precious energy and processor cycles, the function `Sample_Signal_Strength()` instead reactively measures the signal strength values for a window of time after the PDR values fall below a threshold (the threshold we have identified in Section 4.1.3), and returns the maximum value of the signal strengths during the sampling window[2], which is denoted as SS. We note that the duration of the sampling window should be carefully tuned based upon the traffic rate, the jamming model, the measuring accuracy, and the desired detection confidence level.

The function `SS_ConsistencyCheck()` takes as input the maximum PDR value of all the neighbors, denoted as $MaxPDR$, and the signal strength reading SS. A consistency check is performed to see whether the low PDR values are consistent with the signal strength measurements. If the signal strength SS is too large to have produced the observed $MaxPDR$ value, then `SS_ConsistencyCheck()` returns False, else it returns True.

The consistency check may be conducted empirically as follows. During deployment, or during a guaranteed time of non-interfered network operation, a table (PDR, SS) of packet delivery ratios and signal strength values are measured. We may divide the data into PDR bins and calculate the mean and variance for the data within each bin. Or, we may conduct a

[2]In order to prevent spurious readings and have improved stability, in practice we use the average of the top three signal strength readings.

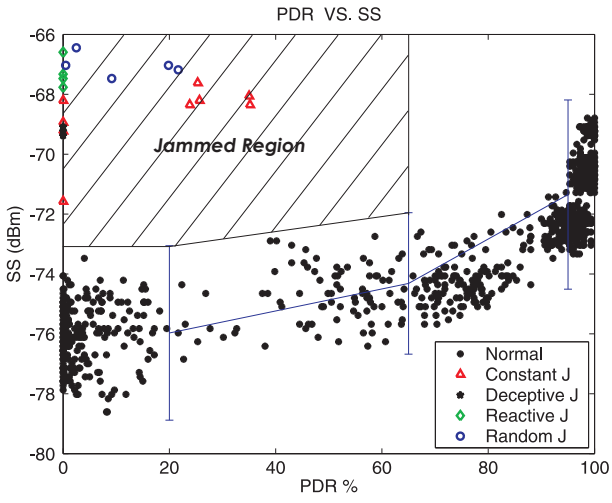


Figure 4.4: The (PDR, SS) measurements, indicating the relationship between PDR and signal strength. Also presented are the (PDR, SS) values measured for different jammers. The data was binned into three PDR regions, $(0, 40)$, $(40, 90)$ and $(90, 100)$, and the corresponding 99% confidence intervals are presented. The shaded region is the jammed-region, and corresponds to (PDR, SS) values that are above the 99% signal strength confidence intervals and whose PDR values are less than 65%.

simple regression to build a relationship between PDR and SS. The output of the binning or the regression is a relationship from which we may calculate an upper bound for the maximum SS that would have produced a particular PDR value in a non-jammed scenario. Using this bound, we may partition the (PDR, SS) plane into a benign-region and a jammed-region.

We conducted an experiment using MICA2 Motes to validate Algorithm 4. We gathered (PDR, SS) values for a source transmitting to a receiver node at a power level of roughly -5dBm. The PDR values were calculated using a window of 200 packets, while the SS values were sampled every 1msec for 200msecs in order to provide sufficient resolution to capture the jammer behavior during a reactive jammer attack. The packets were 33 byte long and transmitted at a rate of 20 packets per second. The source receiver separation was varied in order to produce a full spectrum of normal (PDR, SS) values, as depicted in Fig. 4.4. Using these values, we found the 99% SS confidence bars values for $(0, 40)$ $(40, 90)$ and $(90, 100)$ PDR regions. We depict these confidence bars, and define the corresponding jammed-region to be the region of (PDR, SS) that is above the 99% signal strength confidence intervals and

whose PDR values are less than 65%. The jammed-region is shaded and appears in the upper-left corner of Fig. 4.4. We then performed experiments where we introduced the different jammers. The reactive jammer that we used sent out a 20-byte long interference packet as soon as it detects activities on the channel, while the random jammer had t_j = U[0,31] and t_s = U[0,31]. We varied the source-receiver configurations as well as the location of the jammer, and measured the resulting PDR and SS values. As can be seen in Fig. 4.4, the (PDR, SS) values for all jammers distinctively fall within the jammed-region.

It is to be noted that the jammer in this experiment had a transmission power level of roughly -4dBm, which is stronger than that of the source. In fact, in order for the jammer to be more effective, it needs to operate at a relatively higher power level. However, a jammer using higher power will further decrease the PDR value and increase the SS measurement, thus pushing the resulting (PDR, SS) pair further towards the upper left corner, making it more distinct the benign-region. On the other hand, a jammer that operates on a lower power level is not as effective in interfering with the network operations. As a result, the combination of PDR and signal strength is quite powerful in discriminating a jammed scenario from various network conditions.

4.2.2 Location Consistency Checks

We now discuss a second consistency checking algorithm for detecting the presence of a radio interference attack. Whereas `PDRSS_Detect_Jam` employs signal strength to validate PDR measurements, the `LOC_Detect_Jam` algorithm employs location information. In addition to the assumptions listed earlier, for `LOC_Detect_Jam` we also assume that all legitimate neighbor nodes transmit with a fixed power level, such as the default settings when the sensor or ad hoc network was originally deployed. While this assumption holds for many real network settings, we would like to point out that scenarios where nodes have varying transmission powers can be addressed by easy extensions to our algorithm.

In `PDRSS_Detect_Jam`, the sampling granularity and the window length for measuring signal strength are two parameters that must be carefully set based upon the assumed jammer models as well as the underlying network traffic conditions. As noted earlier, it may not be practical to sample the signal strength with a fine granularity over a long window of time, and

for this reason `PDRSS_Detect_Jam` employs a reactive consistency checking strategy in the sense that signal strength measurements are performed after PDR measurements fall below a threshold.

Instead of employing a reactive consistency check, the `LOC_Detect_Jam` algorithm uses a proactive consistency check. Rather than a node reacting to conduct measurements, the location consistency checking scheme involves information that is already made available to the wireless node prior to determining that PDR values are suspicious. As a consequence of this, the granularity and window length at the detector is no longer an issue. We note, in our specification of `LOC_Detect_Jam` that, although we require each node to transmit a location advertisement message, the issue of window length and granularity of signal strength sampling has been translated from a complicated issue involving assumptions regarding the jamming model into an issue regarding a node's mobility. As shall be seen, the analogous notion of position message frequency may be simply addressed using knowledge of node mobility and an assumption regarding the nominal packet delivery ratio of the network.

The `LOC_Detect_Jam` protocol requires the support of a localization infrastructure, such as GPS [19], or other localization techniques [7, 39, 50], which provides location information to wireless devices. We assume that this localization infrastructure is not able to be attacked or exploited by potential adversaries. Recently, countermeasures have been proposed to protect localization services from being exploited by adversaries [12, 41, 44]. In the `LOC_Detect_Jam` protocol, we again use PDR as the metric indicating link quality. A node will decide its jamming status by checking its PDR and deciding whether the observed PDR is consistent with what it should see given the location of its neighbor nodes. Conceptually, neighbor nodes that are close to a particular node should have high PDR values, and if we observe that all nearby neighbors have low PDR values, then we conclude that the node is jammed.

In our protocol, we let every node periodically advertise its current location and further let each node keep track of both the PDR and the location of its neighbors. Due to node mobility, it is necessary that the location advertisements occur with sufficient frequency to be able to reliably capture the migration of neighbors from regions of high PDR near node A to regions of lower PDR further from node A. If a jammer suddenly comes into the network near node A, then the location information that node A has will correspond to the location of the neighbors

Algorithm:LOC_Detect_Jam
$\{PDR(N) : N \in Neighbors\}$ = Measure_PDR()
$(n, MaxPDR) = (\arg\max, \max)\{PDR(N) : N \in Neighbors\}$
if $MaxPDR < PDRThresh$ **then**
  $P_0 = (x_0, y_0)$ = GetMyLoc()
  $P_n = (x_n, y_n)$ = LookUpLoc(n)
  d = dist(P_0,P_n)
  $CCheck$ = LOC_ConsistencyCheck($MaxPDR$, d)
  if $CCheck == False$ **then**
    post NodeIsJammed()
  end
end

Algorithm 5: Jamming detection algorithm that checks the consistency of PDR measurements with location information.

prior to the start of the interference. Analogous to `PDRSS_Detect_Jam`, if node A finds that the PDR values of all of its neighbors are below the threshold $PDRThresh$, then node A will perform a consistency check by using the position P_n of the neighbor who had the maximum PDR. The distance between P_n and P_0 (i.e. the location of node A) is calculated, and together $MaxPDR$ and d are used as input into `LOC_ConsistencyCheck()` to conduct a location-based consistency check.

`LOC_ConsistencyCheck()` operates in a manner similar to `SS_ConsistencyCheck()`. During deployment, a table of (PDR, d) values are gathered to represent the profile of normal radio operation for node A. As in `SS_ConsistencyCheck()`, we may define a jammed-region and a benign-region using either a binning procedure or regression to obtain lower bounds on the PDR that should be observed for a given distance under benign radio conditions using measured data. If the point $(MaxPDR, d)$ falls in the jammed-region, then the node declares it is jammed.

We note that, just as in the operation of `PDRSS_Detect_Jam`, the assumption that every legitimate node transmits a minimal baseline amount of traffic with which to estimate PDR is paramount to the operation of the `LOC_Detect_Jam` protocol. This baseline amount of traffic may coincide with the transmission of location advertisements in order to reduce the overhead of the protocol. The baseline traffic assumption allows us to declare the PDR to be 0 when no packets are received from a neighbor node within a given time period. This assumption is particularly important for handling scenarios where every neighbor node is jammed, as it allows `LOC_Detect_Jam` to pass into the location-based consistency check, which will allow the algorithm to declare that the node is jammed since its neighbors should have delivered at

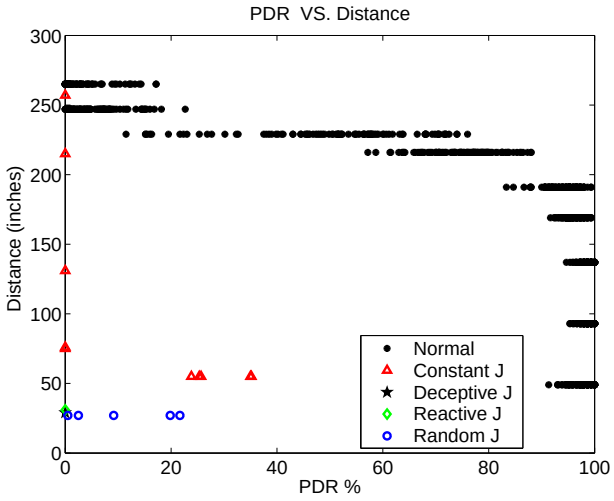


Figure 4.5: The (PDR, d) measurements, indicating the relationship between PDR and distance between source and receiver. Also presented are the (PDR, d) values measured for the different jammer models.

least a minimal amount of packets. Finally, we note that we have disregarded the extremely unlikely event that all neighboring devices have faulted or depleted their power resources.

We conducted an experiment to validate Algorithm 5. The setup of the experiment was the same as the experiment used to validate Algorithm 4. We gathered (PDR, d) values for normal operation as well as for scenarios involving the different jammers, as depicted in Fig. 4.5. As can be seen in Fig. 4.5, the (PDR, d) values for the jammer scenarios, where the source-receiver separation was small, are distinctly separated from normal operation values, and hence fall in the jammed-region. Again, we would like to point out that, for a reasonably dense network where every node has one or more neighbors that are close to itself, a jammer's presence can be easily identified, as shown in Fig. 4.5. If a node, on the other hand, does not have a nearby neighbor, then the PDR of that node, even without the jammer, is rather poor (Fig. 4.5). For these nodes, the effect of a jammer will not be noticeable anyway.

We now address the frequency of node position advertisement. There are two factors that affect the frequency: first, nodes may move towards or away from each other, and second, position messages may be missed, especially for neighbors farther away from node A. We may address the first factor by setting a requirement that a node announces its location whenever it has moved a distance δ from its previous position. By using the device's velocity v, we find that

a device should update its position at least every $\tau = \delta/v$ seconds. To address the second issue, we assume that each device seeks a guarantee of η that its position announcement will arrive to neighbors who are sufficiently close to have at least a nominal packet delivery ratio of q. Assuming independence of successive transmissions of position announcement messages, the cumulative distribution for the amount of transmissions T before the first successful delivery is

$$F_T(T) = 1 - (1-q)^T, \quad \text{for } T \in \{1, 2, 3, \cdots\} \tag{4.1}$$

From the cumulative distribution, we may find the amount of transmissions, $\tilde{T}$, needed to have a guarantee of η that the position announcement will have been heard. Combining the two factors, a node should announce its position every $\tau/\tilde{T}$ seconds. The frequent announcement of position information guarantees that nodes will have knowledge of their neighbor's position.

4.3 Summary and Conclusion

The shared nature of the wireless medium will allow adversaries to pose non-cryptographic security threats by conducting radio interference attacks. In this chapter, we have studied the issue of detecting the presence of jamming attacks, and examined the ability of different measurement statistics to classify the presence of a jammer. We showed that by using signal strength, carrier sensing time, or the packet delivery ratio individually, one is not able to definitively conclude the presence of a jammer. Therefore, to improve detection, we introduced the notion of consistency checking, where the packet delivery ratio is used to classify a radio link as having poor utility, and then a consistency check is performed to classify whether poor link quality is due to jamming. We introduced two enhanced detection algorithms: one employing signal strength as a consistency check, and one employing location information as a consistency check. We evaluated the effectiveness of each scheme through empirical experiments and showed that each of the four jammer models we introduced in Chapter 3 can be reliably classified using our consistency checking schemes.

Chapter 5

Channel Surfing: Defending Wireless Networks Against Radio Interference

In this chapter, we examine the ability of a wireless network to cope with radio interference. We propose the use of *channel surfing*, whereby the sensor nodes adapt their frequency allocations as needed to avoid interference. The challenging research question here is how to establish network connectivity between multiple frequency zones. The inherent diversity in network configuration, interference model, and platform setup suggests that no single solution is sufficient. We examine four strategies to restore network connectivity across multiple channels, each having unique characteristics and advantages. Although channel surfing may be applied to more general wireless networks (e.g. 802.11), in order to validate our strategies, we focused our study on a sensor network platform, and have implemented our methods on a 30-node Mica2 sensor network testbed. During the process of implementation, we have overcome a number of challenges and have demonstrated that all four strategies can maintain network operations in the presence of jamming/interference.

We begin the chapter in Section 5.1 by providing an overview of the sensor network and interference model used in our studies. We next give an overview of channel surfing in Section 5.2 and Section 5.3, where we detail a set of increasingly sophisticated channel surfing protocols. In Section 5.4 and Section 5.5, we describe our validation effort on our sensor testbed.

5.1 System Models

The objective behind this chapter is to examine networking issues associated with adjusting channel assignments in a sensor network in order to avoid radio interference. In this section we outline the basic sensor communication and jamming model that we use throughout this chapter.

5.1.1 Our Sensor Communication Paradigm

There are many choices for sensor platforms and data dissemination models available to the sensor network designer. The broad range of choices implies that there are many different directions that one can take in order to tackle the problem of radio interference. Early on in our studies we found that it was impractical to devise a generic approach that worked across all varieties of sensor networks, and instead found that it was necessary to tailor the design of our solutions to a specific communication paradigm.

Channel surfing requires that sensor radios change their *channel* allocations. In order to commence with channel surfing, the radio devices employed must have a notion of a channel. Most sensor platforms have a natural form of channelization that is accomplished by changing the carrier frequency. For example, in our validation efforts, we use the 916.7MHz Mica2 platform and separate the channels by 800KHz. In the algorithm discussion, we assume that channelization exists.

Another important factor in the sensor communication paradigm is the choice of the data dissemination method and the associated routing protocol. From a sink's viewpoint, there are two main data dissemination paradigms: few-to-one data dissemination (whereby a sink is connected to one source node or a small number of source nodes), as in Directed Diffusion [29] and SPIN [57]; and many-to-one (whereby a sink node is connected to a large portion of a network), as in Zebranet [30] and TAG [46]. In this chapter, we have chosen to focus on the many-to-one model, and we assume that there is only one sink that collects data. For the many-to-one model, our studies focus on tree-based routing schemes, a popular family of routing protocols whereby the network establishes and maintains a forwarding tree [46].

We briefly touch upon the salient features of tree-based routing needed for our discussion. In these schemes, a routing tree is formed with the sink node serving as the root of the tree. A node selects its routing *parent* as its best radio neighbor in the direction of the tree's root. In such a tree-based routing structure, a node usually has a parent (except the sink) and one or more children (except the leaf nodes), wherein it receives data packets from its children, and sends packets to its parent. A node's parent and children are considered its *neighbors*. Besides the parent and children, there may be other nodes that are within a node's communication range.

These nodes, are *not* neighbors of the node. In this chapter, we use the term neighbors to refer to the topology-based relationship rather than a physical location-based relationship.

5.1.2 Our Interference Model

We now turn our attention to describing the interference model. When considering issues of radio interference and jamming, it must be emphasized that there is a very broad range of capabilities that one might assume is available to the interferer, ranging from whether the interferer is incidental or intentional, powerful or resource-constrained, narrowband or broadband, or static or adaptive.

It is immediately apparent that if the jammer is a high-powered, non-coherent, broadband source of interference (e.g. capable of occupying all channels simultaneously), then there is no hope for building a resilient sensor network short of choosing a different PHY-layer transceiver with a powerful anti-jam margin [61, 69]. Further, it should also be noted that an aggressive adversary may jam a single channel (e.g. by reprogramming another sensor) at a time and rapidly switch between channels to effectively disrupt network services across all channels. Both cases represent powerful, aggressive broadband jamming adversaries.

Instead of considering powerful interference models for which the only viable defense might be powerful physical layer techniques, in this work, we consider a non-intentional or a relatively benign adversary in which the interferer blocks one (or even a few) of the channels at a time. This model has been considered elsewhere in the literature [40, 76, 78, 79], and it can be accomplished by employing a narrowband RF source (e.g. such as a waveform generator) or by another sensor node that has been reprogrammed to jam a particular channel. Further, even if the interferer hops to different channels, we assume that the interferer stays on one channel for a brief period of time before switching to another channel. Related to these assumptions, we note that we consider traditional security threats [60, 73](such as authentication, communication confidentiality, and coping with node compromise) to be orthogonal to the issues discussed in this paper.

A jammer, whether incidental or intentional, can significantly affect the network's communication. As we shall point out in the next section, our channel surfing algorithms operate on-demand– when interference is detected, channel adaptation is used. Hence, as the starting

point for coping with jamming, it is necessary to detect jamming. Since spectrum utilization is a local phenomena, detecting the presence of a jammer must be done by each individual sensor node– that is, a node can only detect whether it is jammed, not whether other nodes are jammed. Several jamming detection approaches have been proposed, ranging from measuring simple properties (e.g. ambient signal strength and packet delivery ratio [77, 79]), to more complicated consistency checks. In this chapter, we utilize our detection scheme presented in chapter 4, which involves a consistency checking process on each node to ensure reliable identification of jamming attacks. We note that even if an interferer only degrades the network link quality partially (e.g. by blocking half of the packets), the classification of such a situation is the responsibility of the detection algorithm and the policy the network operator has used to define "interference". The implication of this fact is that in the channel surfing algorithms we describe, we would like for the jamming/interference detection process to be a separate module that is utilized by the channel surfing algorithm. As long as a node is declared "jammed", channel surfing will kick in.

5.2 Channel Surfing Overview

Typically, when radio devices communicate they operate on a single channel. Here, the concept of channel may be a single operating frequency or more generally may be any division of the operating spectrum. For the sake of discussion, we shall assume that the notion of a channel is associated with a single carrier frequency. Channel surfing is motivated by frequency hopping, a physical layer technique used in spread spectrum communication. As noted earlier, we assume that the adversary blasts on only one channel (or at most a few channels) at a time.

The basic idea behind channel surfing is that the link layer channel assignments should be changed in order to avoid interference. When thinking of how to achieve this, a natural idea is to directly apply the philosophy of constantly changing the channels (as is done in frequency hopping spread spectrum). However, employing such a strategy at the link layer, can be detrimental and costly to providing network services. First, if one rapidly changes the channel assignment, then it is necessary to have a fine granularity of synchronization across the entire network. Even if channels are changed less rapidly, for example on the order of a hundred

```
Algorithm: Channel Surfing Framework
while (1) do
    if (NeighborsLost() == TRUE) then
        working_channel = next_channel;
        for (i = 0; i < m; i++) do
            SendInquiryPacket();
            if (ReceiveReplyWithinT() == TRUE) then
                break;
            end
        end
        if (i==m) then
            working_channel = original_channel;
        else
            Use a Channel Surfing Strategy;
        end
    end
end
```

Algorithm 6: The channel surfing framework.

milliseconds, constantly changing the channel incurs switching overhead. For example, routes that exist on one channel may not be guaranteed to exist on other channels, and frequently changing channels may cause the network to become unstable, thereby necessitating frequent route maintenance and discovery each time a channel is changed. Further, such penalties are incurred even if there is no interference presence.

Based on these arguments, a better strategy would be to only adapt channel allocations when needed, i.e. when interference is detected. In order to achieve an interference-resistant sensor network, we propose a collection of distributed on-demand channel surfing algorithms. As the starting point for these algorithms, each node runs a jamming detection process to determine whether it is jammed. In channel surfing, those nodes that detect themselves as *jammed* nodes should immediately switch to another orthogonal channel and wait for opportunities to reconnect to the rest of the network. After the jammed nodes lose connectivity, their neighbors, which we refer to as *boundary* nodes, will follow Algorithm 6 to discover the disappearance of their jammed neighbor nodes (e.g. via a drop in link quality) and temporally switch to the new channel to search for them. If the lost neighbors are found on the new channel, the boundary nodes will participate in rebuilding the connectivity of the entire network.

Before we move onto our specific algorithms, we first discuss a few challenging issues in the above channel surfing framework. The first challenge concerns the potential boundary nodes. The basic scheme specifies that a boundary node should switch to a new channel after its neighbors are jammed and have escaped to the new channel. However, if a node immediately

probes the next channel whenever it experiences a poor link quality with any of its neighbors, the system will enter a non-stable state because wireless sensor networks inherently experience frequent link quality degradations or even topological changes [75, 81]. Fortunately, after carefully studying the working of the underlying system, we found that it is possible for boundary nodes to correctly differentiate jammed neighbors from those neighbors that just had a poor link with the boundary node. This can be explained as follows. For tree-based routing, a node has precisely two types of neighbors/links (one link with its parent, and possibly several links with its children). Thus there are two possibilities for broken links and we can address these each separately. First, if a node witnesses degradation in the link with its parent, then the underlying routing protocol will first attempt to find another, suitable parent node. If the node finds a replacement parent, then it will announce its parent selection in a routing update. However, only if there is no suitable replacement parent, the node will probe the next channel to find its parent. In this way, the node does not need to switch channels if it can still maintain its normal network operations. Next, we cover the case of a node losing one of its children. If the lost child node was not jammed, but just connected to a new parent, the node *should* hear its former child's routing announcement. If it does not witness the child's routing announcement within a specified window of time, then it will probe the next channel looking for its lost child. Thus, a node will become a boundary node either because it is the parent of jammed nodes, or because it is the jammed nodes' child that cannot find a new parent.

After detecting the loss of a neighbor, the boundary node should not switch to the new channel too quickly. If it switches too soon, it may arrive at the new channel before the jammed nodes. To understand this, consider a scenario where the jammer starts interference at time t_0. At that time, the jammed nodes will not be able to send out packets. However, since it takes less time for a node to detect the absence of a neighbor than it does for a node to decide it is jammed, the boundary nodes will detect the absence of a jammed node at time $t_0+\delta_1$, while the jammed node will declare itself jammed at $t_0+\delta_2$, where $\delta_2 > \delta_1$. If the boundary nodes switch to the new channel immediately after $t_0+\delta_1$, they will not find the jammed nodes there. Rather than have the boundary node wait on the next channel, which would prevent it from conducting its primary objective of relaying messages to the sink, or having the node constantly flip-flop between channels looking for its children, we should make the boundary nodes wait for at least

an additional $\delta_2 - \delta_1$ amount of time before switching to the next channel. The values of δ_1 and δ_2 are characteristic of the particular routing protocol, and the jamming detection scheme.

In addition to the switch timing for a boundary node, it is important to have a discovery protocol by which a boundary node can find its neighbors on the new channel. After a node switches to the new channel searching for its neighbors, it should send out an "inquiry" message, such as "Is my neighbor X here?" If it receives a reply from X, it will start working on repairing the connectivity between X and the sink. Otherwise, it waits for time δ to send another message. If the node does not hear from X after a few trials, it assumes the child is not jammed, returns to the original channel and resumes its original operation in the network. In total, the time spent probing the next channel should be less than δ_2 to avoid cascading channel probing.

Finally, it is desirable to choose the next channel so that the adversary cannot predict what channel the nodes will surf to. We may choose to chain the channel selections using a keyed pseudo-random generator [9]. If the n-th channel assignment is $C(n)$, then we take $C(n+1) = E_K(C(n))$, where K is a key shared by all nodes in the network that is used exclusively for channel assignment. If ever $C(n+1) = C(n)$, then the channel assignment proceeds to $C(n+2)$ and so on until a different channel is selected. Finally, if the jammer can block several channels, then after a jammed node escapes to a new channel, it should first detect whether the channel is jammed before it starts working on that channel. In practice, one typically has to check at most a few iterations in order to find a new channel that is not jammed.

5.3 Channel Surfing Strategies

After the boundary nodes discover that their neighbors are jammed and have escaped to another channel, they will attempt to reconnect the jammed nodes with the rest of the network. In this chapter, we propose two different classes of techniques that the boundary nodes can use to repair network connectivity: (1) *coordinated channel Switching*, in which the boundary nodes participate in transitioning the entire network to the new channel, thereby reestablishing the network on the new channel; and (2) *spectral multiplexing*, where the boundary nodes multiplex between the old channel and the new channel, serving as a "bridge" that connects

nodes operating on different channels. In this section, we discuss these two strategies, outline their challenges, and highlight their advantages and disadvantages. Further practical issues are discussed in Section 5.4.

5.3.1 Coordinated Channel Switching

The idea behind coordinated channel switching is rather simple: the entire network must coordinate its evasion of the interference by switching to the next channel and resuming network operation there. These strategies are characterized by a transition phase during which an increasing amount of the nodes switch to the next channel. Following the transition, the entire network resumes stable operation on the next channel. Within the family of coordinated channel switching protocols, the strategies are characterized according to the coordination strategy governing the transition. In this section, we examine two different strategies: first, a technique whereby each node autonomously follows its neighbors to the next channel; second, a strategy whereby nodes are accelerated through the transition phase through the broadcasting of channel changing commands by the boundary nodes.

Autonomous Channel Switching

In the autonomous channel switching scheme, a node will autonomously switch to the new channel if it detects that some of its neighbors have moved to the new channel. In particular, each node is responsible for determining the next channel $C(n+1)$ it should switch to by using $C(n+1) = E_K(C(n))$. The scheme begins with the jammed nodes detecting they are jammed. After a set amount of time, the boundary nodes will notice that they have not received messages from their jammed neighbors. The boundary nodes will then probe the new channel, searching for their lost neighbors. If the boundary node finds a neighbor residing on the new channel, it will stay in the new channel, extending the zone of nodes in the new channel. This process repeats until the entire network reconnects on the new channel. Overall, a network with n hops from the jammer to the boundary of the network, will take n rounds to complete the channel switching.

Algorithm Walk-through: In order to illustrate the autonomous algorithm, let us walk through

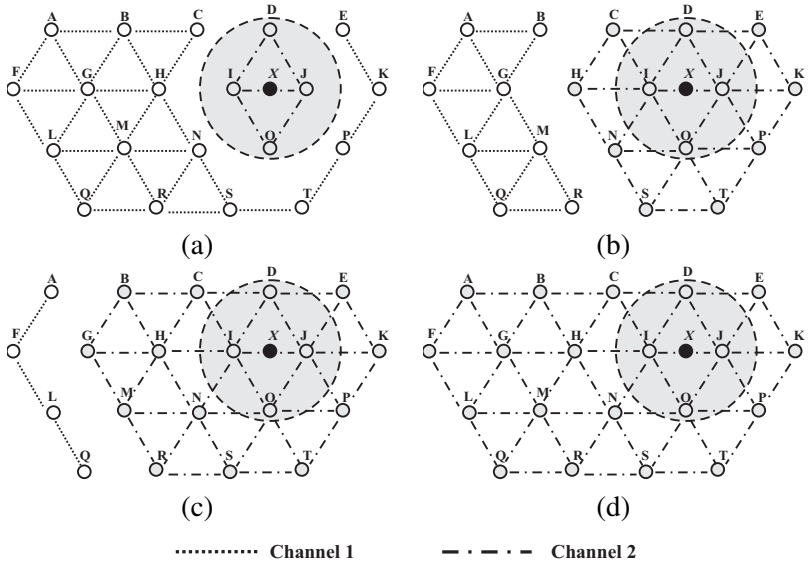


Figure 5.1: A walk-through of autonomous channel switching. The shaded circle illustrates the jammed area, with the jammer X at the center. The entire network switches to channel 2 within four rounds, each round shown in one plot.

the example depicted in Figures 5.1(a)-(d). Let channel 1 be the old channel and channel 2 be the new channel. Here, the jammer X affects nodes $\{D, I, J, O\}$. Upon detecting they are jammed, these four nodes switch to channel 2, as shown in Figure 5.1(a). The dashed-dot lines indicate links that exist in channel 2, while the dotted lines correspond to links in the first channel. The boundary nodes $\{C, E, H, K, N, P, S, T\}$ will notice that their jammed neighbors are no longer on channel 1, and will probe channel 2. When they discover their neighbors on the new channel, they will remain on channel 2 and form a network on channel 2, extending the size of the channel 2 subnetwork. In the next round, the channel 1 neighbors of $\{C, E, H, K, N, P, S, T\}$ will notice that some of their neighbors are missing and will probe channel 2. Upon finding their neighbors in channel 2, they will remain, and the channel 2 subnetwork will grow. This process repeats until the entire network has moved to the new channel. Including the jammed nodes, there are four hops from the jammer to the boundary of the network. Thus, after the jammed nodes evade to the new channel, we need three more rounds for the rest of the network to convene on channel 2, as shown in Figure 5.1(b), (c), and (d).

Algorithm Challenges: This algorithm relies primarily upon the ability of each node to detect

the absence of its neighbors, and to differentiate between whether a node is missing due to fluctuations in link connectivity or from jamming. Hence, when implementing this algorithm, the issues identified in Section 5.2 become very essential to the performance and operation of this scheme.

Discussion: One of the main advantages of this scheme is the simplicity of its description. Further, this scheme introduces minimal additional communication overhead. The nodes in the network merely detect the absence of their neighbors, and probe subsequent channels. Unfortunately, this scheme requires a long latency for the entire network to switch channels. If there are n hops from the jammer to the boundary of the network, then it will take at least $n\delta_2$ time for the network to stabilize (recall, from the discussion in Section 5.2 that we require boundary nodes to wait at least an additional $\delta_2 - \delta_1$ time).

To differentiate between jamming and natural causes of poor link quality, it is necessary for δ_2 to be large. Thus, since the protocol latency is linear in the amount of hops in the network (or roughly square root in the amount of nodes in the network), the issue of scaling becomes a dominant factor for this protocol. As the number of sensors in the network increases, the latency associated with the transition phase can become prohibitive and a large fraction of the sensor data will not be delivered.

Broadcast-Assist Channel Switching

Switching channels autonomously incurs significant latency because (i) a node can only switch after its neighbors have done so, and (ii) it must wait for some time even after its neighbors have switched. Broadcast-assist channel switching addresses both problems. Instead of requiring that every node detect whether its neighbors have switched, a boundary node that has found a jammed neighbor residing on the next channel will facilitate a more rapid phase transition by broadcasting a command. In particular, a boundary node switches back to the old channel, broadcasts a *channel switch command*, and returns to the new channel. Once a node receives this notice, it rebroadcasts the command and switches to the channel specified. Overall, broadcast-assist channel switching facilitates parallel channel switching and, just as parallel execution is usually faster than sequential execution, it can significantly reduce the network

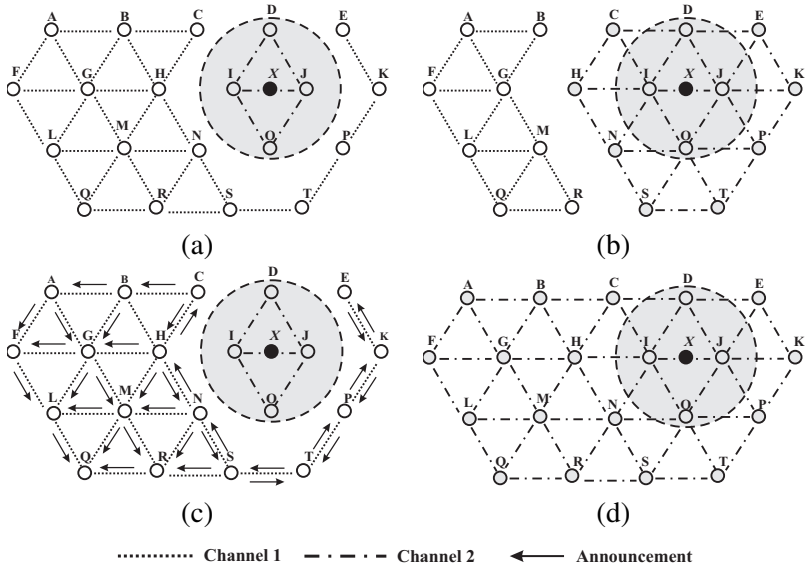


Figure 5.2: A walk-through of broadcast-assist channel switching. The shaded area depicts the jammed area, with the jammer X at the center.

switching latency.

Algorithm Walk-through: We now examine the broadcast-assist channel switch algorithm using the same example as used to describe the autonomous scheme. Nodes $\{D, I, J, O\}$ are jammed by the jammer X and consequently switch to the new channel, e.g. channel 2, as shown in Figure 5.2(a). As a result, nodes $\{D, I, J, O\}$ form the channel 2 subnetwork, and the rest of nodes form the channel 1 subnetwork. After time δ_2, the boundary nodes, $\{C, H, N, S, T, P, K, E\}$ will notice that their jammed neighbors are no longer on channel 1, and will probe them on channel 2 (Figure 5.2(b)). After finding the jammed nodes on the new channel, the boundary nodes return to the original channel temporarily and broadcast a switching notice $(n_{id}, C(n+1))$, where n_{id} is the ID of the sender node, and $C(n+1)$ is the new channel to switch to. The switching notice is sent to the rest of the network through the channel 1 subnetwork, as shown in Figure 5.2(c). The boundary nodes join the jammed nodes on the new channel after broadcasting the notice. Shortly thereafter, the rest of the nodes will switch to the new channel after receiving the switching notice, and reestablish the network on channel 2, as shown in Figure 5.2(d).

Algorithm Challenges: The major challenge facing this scheme is the fact that unreliable/variable

links can cause some nodes to miss a channel switch notice. However, the channel switching command is typically broadcasted independently by multiple boundary nodes. Thus a node is very likely to receive at least one notice, and be able to switch to the new channel. Even should a case arise where a node does not receive a switch notice, it will still autonomously move to the next channel as it will detect that it cannot receive messages from neighbors that have already switched to the new channel.

Discussion: Compared to autonomous switching, the broadcast-assist channel switching incurs much less switching latency. After jamming is detected, the boundary nodes will switch their channel within time δ_2, and then broadcast the switching notice. Suppose the network has n hops between the jammer and the boundary of the network, and that μ is the delay for one-hop transmission ($\mu \ll \delta_2$). Then the overall latency is roughly $\delta_2 + (n-1)\mu$, which is much less than the $n\delta_2$ latency required by autonomous channel switching.

Additionally, the success of performing a broadcast-assist channel switching doesn't depend on the likelihood that each individual node can detect the loss of its neighbors but, rather, as long as one of the boundary nodes finds its lost neighbors in the new channel and informs the rest of network, the network will resume its connectivity in the new channel in spite of the radio interference. Finally, we note that the broadcasted channel switch command should be authenticated [10, 56, 57, 74, 83] to prevent malicious message injection by an adversary. Thus, in practice, the channel switch notice has the format of $E_{K_A}(n_{id}, C(n+1), nonce)$, where K_A is a network-wide authentication key that is distinct from the key used to generate $C(n+1)$, and $nonce$ is included to prevent replay attacks.

5.3.2 Spectral Multiplexing

Performing a coordinated channel switch requires the entire network to reestablish the routing tree as the link connectivity will not be the same on the new channel. The global nature of coordinated channel switching can be a source for significant network cost, and a natural alternative is to employ a local response where only jammed nodes switch channels, while non-jammed nodes remain on the original channel. To guarantee the communication between these two frequency zones, boundary nodes have to work on both channels by repeatedly switching back

and forth between two channels to *relay* packets, a process we call *spectral multiplexing*.

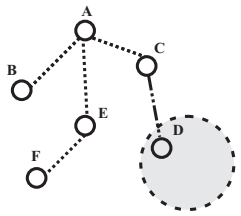
Initially, for spectral multiplexing, jammed nodes that were originally on channel $C(n)$ will switch to $C(n+1) = E_K(C(n))$. However, for spectral multiplexing, the primary challenge lies in the fact that the boundary nodes must carefully decide when they should stay on which channel, and for how long, so that they can minimize the number of packets that cannot be delivered due to the frequency mismatch between the sender and the receiver. If the boundary nodes are configured with dual radios, this scheduling is unnecessary. However, commercial sensor platforms including Berkeley motes only have one radio interface and, as a result, a node can work on only one channel at a time. It is therefore crucial to make sure that the sender and the receiver are able to work on the same channel when they want to exchange messages. Towards this end, boundary nodes must transmit an announcement to its neighbors that they are operating in a "dual-mode" on two channels. Further, these boundary nodes must employ a synchronization mechanism to coordinate the spectral schedules of the sender and the receiver when one party needs to work in "dual-mode". The overall scheduling objective is to ensure that dual-mode nodes are present on the right channel when the neighbors on that channel are ready to transmit[1].

In general, there are two ways of coordinating schedules from different entities: one is to have all the entities adopt synchronous schedules, and the other is to operate in an asynchronous fashion. We thus propose the corresponding methods to coordinate the frequency schedules for neighboring nodes: (1) Synchronous Multiplexing, in which all the nodes share the same schedule by dividing the global time axis into different slots and assigning one slot to a channel; and (2) Asynchronous Multiplexing, in which a node operates on a local schedule, and the boundary nodes make local decisions about when to switch channel.

Synchronous Spectral Multiplexing

In synchronous spectral multiplexing, the entire network is governed by one global clock. The global time axis is divided into slots, and multiple slots form a round. The number of slots in a round is determined by the number of channels the network has to operate on at any specific

[1]The need for scheduling transmissions has also been considered in the context of duty cycling, as in S-MAC [80], in order to preserve energy.



(a) the network topology

slot 1	A, B, C, E, F
slot 2	C, D

(b) the global schedule

Figure 5.3: Illustration of the synchronous spectral multiplexing algorithm.

time. (In this chapter, we limit our discussions on situations where the network works on 2 channels simultaneously, and the discussion can be easily extended to cover situations with more than 2 channels.) Each slot is assigned to a single channel, and during that time slot, the network may only use the corresponding channel– regardless of whether they are jammed, boundary nodes, or not. At the end of a time slot, the entire network utilizes the next channel and, again, the nodes that are not using the next channel do not transmit, nor must they switch channels unless they are dual-mode boundary nodes. By following this global schedule, we can avoid frequency mismatch between a pair of communicators.

Algorithm Walk-through: Figure 5.3(a) presents an example network scenario in which D is jammed, and switches to channel 2. Its parent node, C, thus becomes a boundary node, and has to multiplex between two channels. The rest of the nodes continue to work on channel 1. The global schedule for this case is shown in Figure 5.3(b), which has two slots for each round, with slot 1 allocated to channel 1, and slot 2 to channel 2. Following this schedule, during slot 1, nodes $\{A, B, E, F\}$ work as normal. Node C sends out packets to its parent A, but does not receive any packets from D. At the end of slot 1, these nodes stop their activities on channel 1, and node C switches to channel 2. During slot 2, the only transmitting node is D, and C buffers all the packets it receives from D. At the end of slot 2, D ends its transmissions and C switches to channel 1. These two slots keep alternating in this fashion until the radio interference ends, or for the lifetime of the network.

Algorithm Challenges: There are several challenging issues associated with this scheme: (1) How to synchronize the schedules of every node, (2) how to start multiplexing, and (3) how to determine the slot duration?

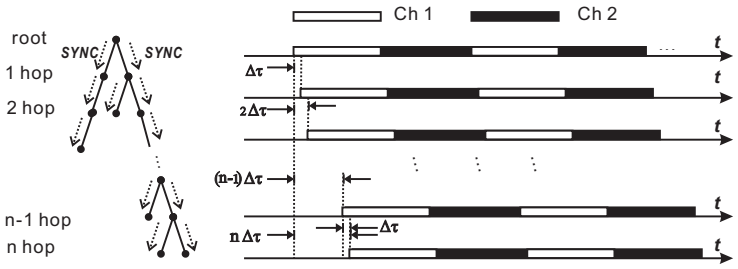


Figure 5.4: Illustration of the synchronization mechanism.

Synchronization: One natural synchronization approach is to have the entire network work under a global synchronized clock, wherein each node maintains a unique and global timescale. Many protocols can be used to establish a global timescale across the entire network, e.g. TPSN (Timing-sync Protocol for Sensor Networks) [22]. A closer investigation, however, reveals that perfect synchronization across the entire network is not only inefficient, but also unnecessary. Instead, since communication takes place locally amongst neighboring nodes, we can focus on achieving a fine synchrony within any local region. Additionally, instead of employing traditional pare-wise synchronization, we let the root initiate the synchronization process by broadcasting SYNC packets to its children.

Nodes use a Timer to start/end a slot; at the beginning of a slot, a node sets its Timer to the duration of a slot, and when the timer expires, a node switches to the next slot. Without periodic synchronization, the timers on different nodes may drift significantly. To avoid this, $SYNC$ packets are sent periodically at an interval much larger than the slot duration. Upon receiving a $SYNC$ packet, a node will immediately terminate the current slot and start a new slot by resetting the Timer to the slot duration. This simple protocol can effectively minimize the synchronization error between a pair of neighbors. The resulting synchronization error, $\Delta\tau$, only includes delays involved in sending, propagating, and receiving $SYNC$ packets.

Building a "global" synchronization from a local synchronization protocol requires a starting reference point that initiates the synchronization process. In a tree-based forwarding structure, it is natural to choose the root of the tree as the reference point. Therefore, in the first round the root first sends out $SYNC$ packets to its children, whose depth is 1, and whose clock will be synchronized within $\Delta\tau$ of the root's clock. Similarly, in the $(i+1)^{th}$ stage, the nodes

with depth i send $SYNC$ packets to their children. Finally, after every node receives a $SYNC$ packet, for a tree with depth of n, some leaf nodes will be $n\Delta\tau$ behind the root. This synchronization procedure is presented in Figure 5.4. After synchronizing, each node will start a new slot upon the expiration of its timer. However, in order to compensate for the synchronization error between its time and its neighbor's time, we require that a node wait for a short, random time period prior to transmission.

Further complicating synchronization is the fact that it must be maintained when nodes work on different channels (coping with drift while multiplexing). For example, in the case where the parent node is on channel 1, while the child is on channel 2, the $SYNC$ packet from the parent will be lost. To address this complication, the $SYNC$ packet should specify which channel the current slot is associated with. In order to guarantee that nodes on both channels are synchronized, the node should send these $SYNC$ packets in rapid succession across both channels. This mechanism ensures the $SYNC$ packets will reach the destination.

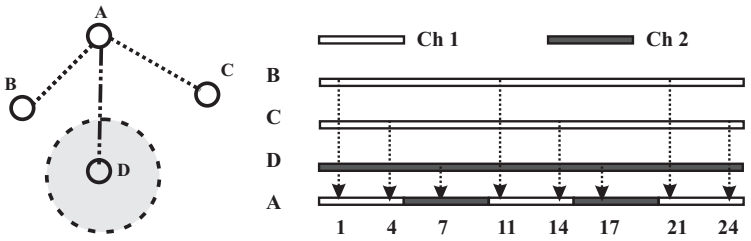
Initiation: As soon as a boundary node discovers that jammed nodes have evaded to the new channel, it will send a message to the entire network on the original channel that contains a list of the channels it will be working on. After a node receives this message, it will compare its own channel with the channel list included in the message, and append its channel if it is not already in the list. In this way, when the message reaches the root of the routing tree, it will contain all the channels the network has to operate on, and the root will create a slotted channel schedule based on this list, and broadcast the schedule down the tree, along with the clock synchronization packets.

Slot Duration: Slot duration is an important parameter in the synchronous spectral multiplexing algorithm. At first glance, it seems intuitive that a shorter slot duration is more desirable because, if a node stays on one channel short enough, the required buffer space will be smaller and, more importantly, less latency would be incurred. However, we found that a smaller slot can be problematic as well, mainly due to the overhead associated with switching channels. Before a node switches to a new channel, it has to complete receiving all the packets that are in transmission. In order to guarantee this, after the Timer expires, we let each node wait for a small amount of time for all the possible transmissions to complete. In our implementation, this

translates into the parent node waiting a little longer because the receiving side is usually the parent node in a tree-based routing. Another problem with short slot durations is that proportionately the synchronization errors will be relatively large with respect to the duration of a slot, thereby affecting this scheme's efficiency. Finally, we note that there is a radio startup cost associated with switching channels(e.g. 250msec for the CC1000 radio chip in the Mica2 mote). Overall, a good slot duration should be determined based on several factors: the available buffer space, the traffic rate, the required message latency, and the synchronization error.

In this study, we choose to adopt the largest slot durations that can satisfy the available buffer space constraints, and we consider our underlying sensing application model to have a periodic traffic pattern. Further, we have empirically witnessed that boundary nodes are more likely parent nodes of the jammed nodes (children of jammed nodes typically look for alternate parents on the original channel), and thus boundary nodes will merely receive on channel 2, but will both receive and send on channel 1. Specifically, based on the traffic rate from each channel and the buffer size, each boundary node calculates the longest stay time it can have on each channel (usually a node should stay on each channel for the same amount of time). In order to understand the calculation, let us look at an example. Suppose a boundary node A has a buffer that can support 10 slots, and it has 1 child on channel 1 that produces 20 packets per second, and 2 children on channel 2 each of which produces 10 packets per second. A can at most stay on each channel for 250 msecs. By spending 250 msecs on each channel, it will receive 10 packets in a round (5 from each channel), which will fill up its buffer. After each boundary node independently calculates a slot duration, the sink will collect all the information, chooses the smallest one as the global slot duration and announces this.

Discussion: Synchronous multiplexing adopts a deterministic global schedule that governs the channel assignment of every node in the network. The deterministic nature of this algorithm guarantees that it can work well even under complex scenarios where multiple nodes need to work on multiple channels and these nodes are neighbors of each other. However, in order to achieve this, every node in the network must pay the extra overhead needed to maintain synchrony between nodes.



(a) the topology (b) the round-robin schedule

Figure 5.5: Illustration of the round-robin asynchronous spectral multiplexing algorithm.

Asynchronous Multiplexing

In the asynchronous multiplexing algorithm, a node is only aware of its neighbors' channel information, but not the channel information of a remote node. The simplest spectral scheduling method is to have a boundary node flip its radio frequency between two channels in a round-robin fashion. However, a completely random round-robin multiplexing strategy ignores the schedules of the communicating parties, and would thus fare poorly. For example, suppose a jammed node, working on the new channel, sends packets at times 10, 20, 30, 40, and 50. If the corresponding boundary node stays on the new channel during time windows [1, 6], [13, 18], [25, 30], [37, 42], [49, 54], then it will miss the packets sent at 10, 20, and 30. The resulting packet loss ratio for the jammed node is now as high as 60%. The above example illustrates the limitation of a random round-robin scheme and highlights the need for some level of coordination between the boundary node and its neighbors for the asynchronous multiplexing scheme.

Algorithm Walk-through: Figure 5.5 illustrates the idea behind the asynchronous multiplexing scheme. In this example, the boundary node A has to receive packets from three nodes, B, C, and D, with the first two working on channel 1 and the last one working on channel 2. Suppose all three nodes send packets every 10 seconds, starting at time 1, 4, and 7 respectively. In this case, starting from time 0, A decides to stay on one channel for 5 seconds and then switches to the next channel for 5 seconds. In this way, A can receive every packet from its neighbors.

Algorithm Challenges: The challenges associated with this schemes include synchronization and slot duration.

Synchronization: To coordinate the schedules of a boundary node and its children, we have adopted a simple protocol that involves the boundary node announcing its schedule (the duration it will stay in the each channel) by notifying its children just after it switches to a new channel. In the example in Figure 5.5(a), A notifies nodes B and C of its schedule as soon as it switches to channel 1, so that they can start transmissions when A enters channel 1, and stop transmissions after A leaves channel 1. Similarly, it also notifies D whenever it is on channel 2. We note that a child node must buffer both its own packets as well as packets coming from its own children while waiting for the dual-mode parent to return to the channel it is working on. To counteract the possibility that the notifications could be lost, a child should start to send its buffered packets immediately after it hears from its parent.

Slot Duration: Determining the slot duration in the asynchronous spectral multiplexing is easier than in the case of the synchronous spectral multiplexing, because in the former situation, not only can different boundary nodes employ different schedules, but they can also stay for a different amount time on each channel. For example, considering the boundary node usually is the parent of the jammed nodes, the boundary node should stay longer on the channel 1 than channel 2, as it has to forward all the packets received in both channels to its parent via channel 1.

Due to the nature of asynchronous spectral multiplexing, nodes can determine their slot durations in a more flexible fashion. Suppose a boundary node decides to stay on channel 1 for t_1 time and channel 2 for t_2 time ($t_1 \geq t_2$), where t_1 and t_2 are chosen according to the traffic volume on each channel and its buffer size. For example, it can choose to have $\frac{t_1}{r_1} + \frac{t_2}{r_2} = B$ where r_1 and r_2 are the traffic rates on the two channels respectively, and B is the buffer size. After setting this baseline schedule, the boundary node can adapt its switching rate as a response to varying network conditions (e.g. topology change, traffic rate change, etc).

Discussion: Compared to synchronous multiplexing, asynchronous multiplexing does not maintain a global schedule, and thus incurs less synchronization overhead. The advantage of asynchronous multiplexing, however, is more pronounced when the jammed region is small and regular. For larger jammed areas, we will have more boundary nodes that work on multiple channels. In this case, the overhead gap between synchronous and asynchronous techniques

Figure 5.6: Our Mica2 testbed consists of 30 motes that are placed on the floor. Nodes are roughly separated from each other by 2.5 feet. The sink is located at the bottom of the figure, with a programming board attached to it.

lessens. A final advantage of the asynchronous method its ability to adapt to local traffic and buffer conditions.

5.4 Sensor Testbed and Metrics

We now focus our discussion on our experimental validation efforts. We have built a 30-node mote testbed, and have conducted numerous experiments with the testbed to evaluate the effectiveness of the four channel surfing strategies in providing interference-resistance.

5.4.1 Testbed Configuration

We have built our sensor network testbed using 30 Mica2 sensor motes. These devices each have a $902 - 928$ MHz Chipcon CC1000 radio. We used 916.7MHz as the original channel and separated our channels by 800KHz, effectively giving us 32 channels. The operating system running on each mote was TinyOS version 1.1.7 [4]. We attached one of the motes to a MIB510CA programming board in order to act as the network sink. In order to conduct experiments that exhibit repeatable characteristics, we chose an indoor laboratory area where we could fix the deployment across the experiments, as illustrated in Figure 5.6. However, due to our space limitations (there are walls just beyond the boundary of the picture), we were forced to reduce the radio range of each mote: in the depicted configuration, we have a node separation of roughly 2.5 feet. We tuned the transmission power of each mote down to -5dBm in order to

restrict the radio range of each sensor node and to increase the network hop count.

The primary objective of this thesis is building a jamming-resistant sensor network, and we have focused our efforts on exploring how a networked system can maintain connectivity in the presence of jamming/interference, regardless of the type of the application data. As a result, we have not attached any specific sensors to the motes. Rather, we modified the Surge application, which comes with TinyOS, to convey experimental statistics. By default, Surge uses a tree-based routing algorithm (which we shall call $SP(t)$) for a single network sink, as detailed in the `MultiHopRouter.nc` file in the TinyOS 1.1.7 release. Since our focus was on networking issues associated with jamming resistance, we did not employ message acknowledgements or retransmissions. In addition to this basic communication model, we note that each sensor message contains a sequencing field (in the routing header) that can be used to estimate performance statistics, such as link quality. Finally, we note that our packet size was 32 bytes, and that a node can buffer at most 24 packets (across all channels).

5.4.2 Implementation of a Sensor Network

Before we could develop the proposed channel surfing strategies on the testbed, we had to modify the existing TinyOS code to address a number of implementation-related issues. In this section, we list a few of the more relevant issues.

The first challenge is devising a reliable link quality estimation technique as this directly affects the efficiency of the routing process. A good estimation technique must be both accurate and resilient to fluctuations. For example, our routing protocol maintains the tree-based topology based on the measured link quality: a node chooses the neighbor that has the best link quality as its parent. Similarly, if a node observes low link quality from the current parent, it will attempt to choose a new parent. On the other hand, due to the low-power and low-fidelity nature of the Mica2 radio, the resulting wireless link quality usually fluctuates greatly, which makes measuring link quality a challenging task.

The existing estimation method utilizes the windowed mean with exponentially weighted moving average estimator, where link quality is defined as:

$$\theta_{new} = \frac{N_{recv}(t)}{\max(N_{exp}(t), N_{recv}(t))} \tag{5.1}$$

$$\theta = \theta_{old} \times \alpha + \theta_{new} \times (1 - \alpha). \tag{5.2}$$

Here θ_{new} is the currently estimated value, $N_{recv}(t)$ is the number of received packets within time t, and $N_{exp}(t)$ is the number of expected packets within time t. In Equation 5.2, the value of α governs how much the past estimation affects the current estimation. The default value for α is set to 0.25 in TinyOS 1.1.7, but we found that this value is too low and the estimated link quality was overly variable. Therefore, the routing topology changes frequently and was hard to stabilize. Through our experiments, we found that $\alpha = 0.75$ yields much better estimates, which incidentally agrees with the value recommended in [75].

After adopting suitable parameters for estimating link quality, we faced the challenge of choosing a parent node. A node chooses its parent based on two criteria: the hop count from the sink and the estimated link quality. Ideally, the parent node should have the smallest hop count and best link quality. However, in reality, nodes that satisfy both criteria may not exist, and we have to prioritize them. In our implementation, we give more importance to link quality than to hop count. For example, we specify that a parent node must have a link quality better than 75% (compared to 10% in the original $SP(t)$). At the same time, we avoid frequent parent switching by adopting the rule that a node can only choose a new parent when the resulting link quality is at least 20% better than the quality of the current link.

In the TinyOS 1.1.7 release of Surge, the application attempts to push packets to the network queue at a fixed rate. However, if the send operation of the previous Surge packet is incomplete, i.e. the `SendDone` has not been received, then the latest packet is discarded. This posed a problem for us as it did not guarantee that a fixed amount of traffic packets would reach the network buffers. In order to guarantee a nominal data rate, we modified Surge to push data regardless of whether the `SendDone` callback was received.

5.4.3 Building a Jamming-Resistant Network

After preparing the underlying sensor network testbed, we next implemented the channel surfing framework and the four strategies. In the implementation of these strategies, we had to address the following issues.

First, we modified the buffering mechanism to address the fact that buffered packets may

need to be sent on different channels. As a result, for each buffered packet, we associated it with an identifier indicating the channel on which it is to be transmitted.

The second issue we addressed was related to the replacement policy of the mote neighbor table. Each Mica2 mote maintains a neighbor table recording the link quality between each node in its radio range (e.g. *radio neighbors*) and itself. Though a node may have many radio neighbors, especially in a dense deployment, the motes in our testbed only have 16 entries in their neighbor table. As a result, an appropriate buffer replacement policy must be employed to sort a node's radio neighbors. The default replacement policy in TinyOS sorts the radio neighbors based on the link quality between each radio neighbor and the considered node. This policy, however, must be modified to implement channel surfing strategies. To understand this, suppose node A's child, B, is jammed. A is supposed to find the link quality between B and itself has degraded, and then probe the next channel to search for B. However, since B is jammed, the estimated link quality between A and B may be so low that B is evicted from A's neighbor table. In this case, A loses awareness of the existence of B, and will not look for it on the new channel. In order to address this problem, we modified the replacement policy so that it always sorts a node's topological neighbors ahead of non-topological radio neighbors. Hence, we can ensure that a jammed node stays in its former parent's neighbor table until the former parent finds out that it is jammed.

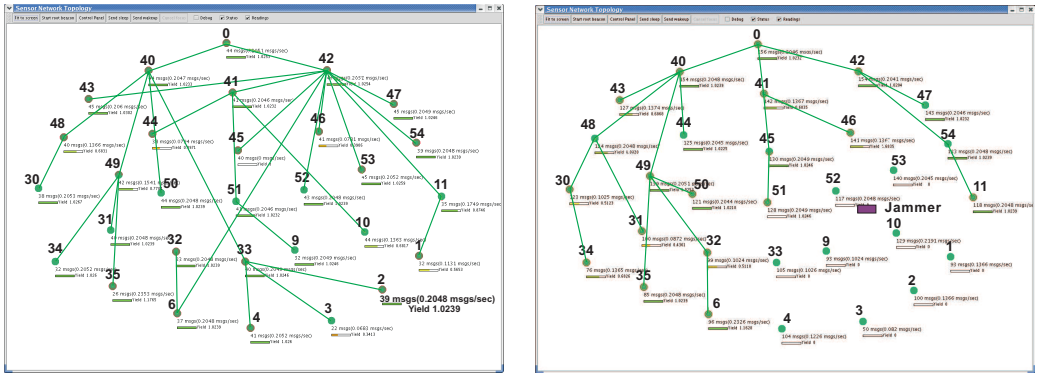
The third issue is that it is necessary to revisit the problem of link quality estimation, and make some further modifications. Estimating the link quality involves comparing the number of packets a node receives with the number of packets it expects by using the sequence numbers of the packets. However, when we operate on multiple channels, this estimation mechanism may cause problems because a dual-mode node will have roughly a 50% link quality, and thus nodes on both channels will not choose the dual-mode nodes as their parents, hindering the realization of spectral multiplexing. We address this by assigning independent sequence numbers on different channels, so that the estimated link quality for a dual-mode node on both channels is sufficiently high.

5.4.4 Performance Metrics for Channel Surfing

The overall impact of channel surfing strategies on a network can be examined in two aspects: the additional overhead it may introduce when the network is in its normal state (i.e. no jamming or radio interference), and the benefit it may have when the network experiences radio interference. To minimize the protocol overhead, we designed the channel surfing strategies so that the protocols do not rely on extra beacons or messages. Instead, we reuse the beacons and the link estimation methods employed in the routing layer. Thus, if the network is in a normal state, it will have comparable performance regardless of whether it employs channel surfing or not. Meanwhile, if the network undergos some form of radio interference, channel surfing can immediately take effect to repair network connection. Finally, a gray area in which channel surfing may impose unnecessary overhead is caused by possible false positives, when channel surfing reacts to neighbor losses due to non-interference reasons. Nonetheless, as pointed out earlier in Section 5.2, we minimize the likelihood of having false positives by only checking a lost neighbor in the new channel under very few circumstances and by adopting a large time threshold. As a result, we believe the proposed channel surfing strategies are both effective in interfered situations and non-intrusive in normal situations. To demonstrate these points, we choose to use the following two performance metrics:

- *Network Recovery:* The main objective of channel surfing is to repair normal network functionalities in the presence of jamming or radio interference. Therefore, we measure their merit by comparing network characteristics in the following three scenarios: (1) under normal conditions, (2) under jamming but without channel surfing, and (3) after applying channel surfing strategies. Specifically, we focus on two network characteristics: network performance (number of packets delivered to the sink), and the latency required to recover network performance.

- *Protocol overhead:* Another class of metrics are related to the overhead introduced by the channel surfing strategies. One obvious concern is that nodes may unnecessarily switch channels. As a result, we have measured the number of channel switches every node experiences throughout the duration of an experiment.



(a) (b)

Figure 5.7: The network topology for jamming experiments. (a) Prior to introducing the jammer, (b) shortly after the jammer is introduced.

We recorded the packet delivery statistics at the sink by using the sequencing field to maintain a running record of the packets that the sink receives from each sensor. Additionally, as noted earlier, we utilized the sensor messages themselves as a means to measure experimental statistics, such as the operating channel for each sensor node.

5.5 Experimental Results

5.5.1 The Impact of Jamming/Interference

We deployed the testbed as illustrated in Figure 5.6, and the resulting tree-shaped routing topology (Figure 5.7 (a)) was captured using the Surge Network Viewer. In the routing tree, the root node, node 0, corresponds to the network sink. We then conducted an experiment to study the network behavior under normal indoor conditions as well as the impact that a single jammer can have on the network. In this experiment, we first let the network run for more than 20 minutes prior to introducing a jammer. For our jammer, we used the same device as legitimate nodes, thus the jammer can only jam one channel at a time. In particular, we used the constant jammer of [78], which is a mote that bypasses the MAC-layer to continually transmit random bits into the network. Figure 5.7 (b) shows the location of the jammer, and the fact that the jammer destroyed the connections between several nodes and the sink. Specifically, in this example, nodes $\{9, 10, 33, 52, 53\}$ lost their connections because they were directly affected by

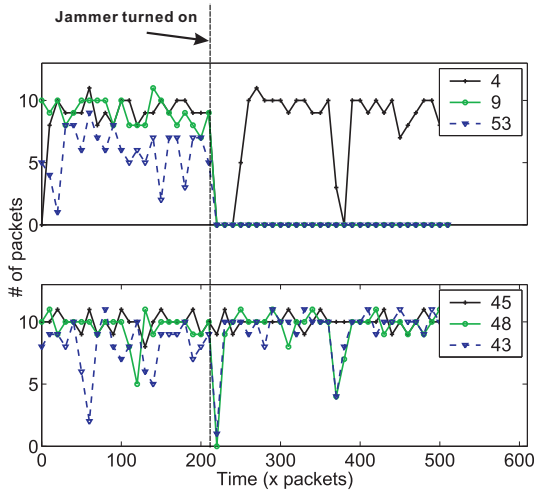


Figure 5.8: Packet delivery time series for a 50-second time window under first normal and then jammed network conditions.

the jammer, while nodes $\{1, 2, 3, 4\}$ lost their connections because their parents were jammed.

We next take a closer look at how the packet delivery quality between each node and the sink evolved with time, both with and without jamming. We present time series for the number of packets delivered to the sink in a window of 10 packet from six randomly chosen nodes in Figure 5.8. In these results, each node generated and sent packets at a rate of 1 packet every 5 seconds. Under a perfect network condition, we expect each node to deliver 10 packets in a 10-packet window, but even under normal network conditions prior to jamming, the traces exhibit short-term temporal fluctuations.

We note that, after introducing the jammer, nodes 9 and 53 were not able to deliver packets to the sink. As discussed earlier, that is because either these nodes were jammed, or all their possible parent nodes were jammed. As a result, the packet delivery ratio became 0. We note that although node 4 lost its former parent due to jamming, it later found a replacement parent on channel 1 and hence its packet delivery became normal after a short interruption.

5.5.2 Coordinated Channel Switching Results

The first set of channel surfing results that we provide are for the two coordinated channel switching protocols described in Section 5.3.1, in which the entire network changes its operating channel to a new channel.

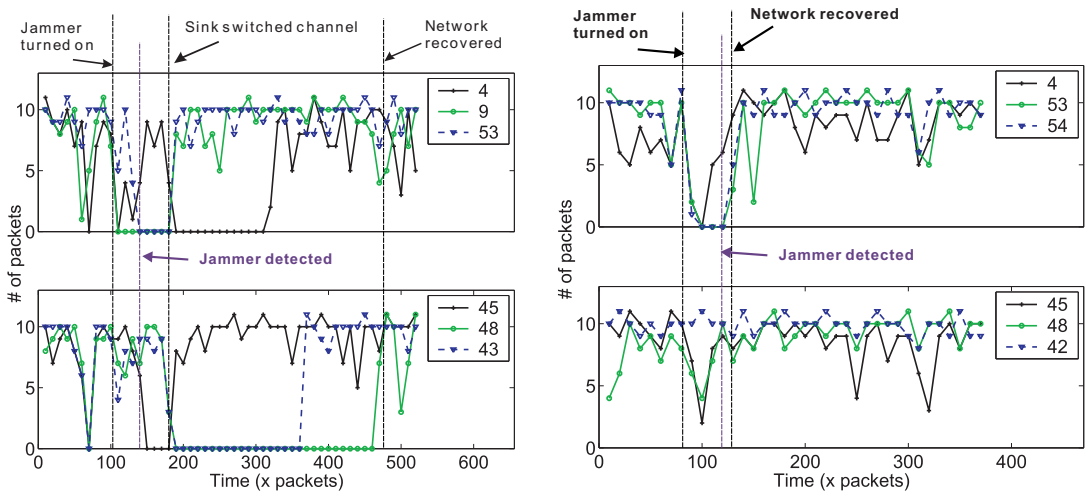


Figure 5.9: (a) Packet delivery time series for a 50-second time window for the autonomous channel surfing strategy. (b) Packet delivery time series for the broadcast-assist strategy.

Autonomous Channel Switching

We conducted an experiment to study how autonomous channel switching repairs a jammed network. In this experiment, we introduced the jammer after the network had run for 10 minutes. As soon as jamming was detected, the autonomous channel switching strategy worked to repair the network topology so that all of the nodes switched channels, forming a new routing tree on channel-2 (not illustrated due to space limitations).

The packet delivery time series for the experiment are presented in Figure 5.9. This plot presents the time series of number of packets delivered to the sink in a 10-packet window, from nodes $\{4, 9, 53, 45, 48, 43\}$. These six nodes represent some jammed (i.e. 9 and 53) and unaffected nodes. The autonomous algorithm has different impacts on different nodes, depending on their locations in the topology:

1. Before the sink switches to channel 2, those nodes that are not affected by the jammer will still continue delivering packets on channel 1, thus maintain good delivery ratio. Please refer to the traces for nodes 4, 48, and 43 between the 100th interval (when the jammer was turned on) and the 174th (when the sink switched channel).

2. Nodes that switch channel before the sink, either because they are jammed (i.e. node

9) or because they detect the loss of their neighbors before the sink (i.e. node 45), are able to resume packet delivery as soon as the sink switches. Please refer to the traces for nodes 9, 45, and 53 after the 174th interval, when the sink switched channel. Of course, these nodes usually suffer disrupted network services before the sink switches.

3. After the sink switches to channel 2, those nodes that were working on channel 1 suffer from low delivery ratios until they detect some of their neighbors have evaded, and they decide to switch. For example, nodes 4, 48, and 43 take a long time to reconnect to the sink. Among these three nodes, node 4 switches to channel 2 faster than the other two because it is closer to the nodes using channel 2.

Looking at traces from different nodes in the network, we can conclude that how fast a node adopts a new channel in a jammed scenario depends on several factors: (1) whether the node is jammed or not; (2) whether the node switches channel before the sink; and (3) if the node was unaffected by the jammer, its topological distance from the sink. We note that, if the percentage of nodes that are directly affected by the jammer is larger, then the network can switch to a new channel faster. Otherwise, it may take a long latency for some nodes to join the rest on the new channel. In this particular example, node 48 required 290 packet intervals to reconnect after jamming occurs.

Another important statistic is how many times a node switches channels throughout the experiment. A good channel surfing scheme should try to minimize this amount. Autonomous channel switching incurs only 1 channel switch for every node. This is due to its simplistic nature as well as the assurance that a node does not probe the next channel too soon. The latter is guaranteed by the following two facts: (1) in the case of a disappearing child, the underlying routing infrastructure can help a node differentiate a child that has evaded to the next channel from a child that has chosen another parent node; and (2) in the case of a disappearing parent, a node simply chooses another parent without probing the next channel, and only probes when there is no legitimate parent candidate.

Broadcast-assist Channel Switching

Next, we conducted experiments to study the effectiveness of the broadcast-assist channel switching method. Just like autonomous channel switching, we observed that the broadcast-assist channel switching method completely restores connectivity for every sensor to the network sink on the new channel. As expected, compared to autonomous channel switching, broadcast-assist channel switching can significantly reduce the transition latency for the entire network to escape to the new channel, as indicated by the packets-delivered time series shown in Figure 5.9. Here we selected a sampling of six nodes, and observed that regardless of a node's position, they can resume operations on the new channel quickly and almost at the same time. The switching latency is roughly the sum of jamming detection latency, probing time, and the broadcast latency. Specifically, the transition phase only took 46 packet intervals, and 39 out of the 46 intervals were used for the jammed nodes to detect they are jammed as well as for the boundary nodes to find out their children nodes are missing. We would like to emphasize that we purposefully let a node wait for a rather long time period (i.e. 39 packet intervals) before probing the next channel after it detects a poor link quality between itself and its children. *We take the viewpoint that sensor networks will experience a LOT more temporary topological changes than longer-term jamming/interference, and that we therefore must reduce the false positives of channel probing to achieve more stable network operations.* Also, we would like to point out that even with such a conservative approach, we could improve the recovery process by having jammed nodes buffer packets during the network transition periods.

We also measured the total number of channel switches for these six nodes versus time. As expected, we saw that, prior to introducing the jammer, no nodes switched channels because our algorithms were tuned to have very low false positives when determining whether to probe next channel. As soon as the jammer started, since nodes 53 and 54 were directly affected, they switched channels, and stayed there afterwards, thus switching only once. Node 42, however, reported 3 channel switches because it was a boundary node that first switched to channel-2 probing its child, then switched back to the original channel to broadcast the switch notice, and finally switched back to the new channel to resume network operations. Other boundary nodes exhibited similar behavior, while more distant nodes only switched once as a response to the

switch notice. Overall, the broadcast-assist channel switch incurs very few channel switches for every node. Finally, we note that we conducted experiments with multiple simultaneous jammers in different positions, and observed that the broadcast-assist method was able to repair the network in these cases with latencies on the order of 50 packet intervals.

5.5.3 Spectral Multiplexing Results

We now examine results for spectral multiplexing. In these methods, we only switch channels for the jammed nodes while a subset of nodes multiplex between the original and new channels. We note that it is unfair to compare coordinated channel switching with spectral multiplexing under the same network configuration because these two strategies are complimentary to each other, each designed to deal with different situations. Specifically, the coordinated strategy is suitable for cases where a large region of the network is jammed, while the multiplexing strategy is suitable for cases with much smaller jammed regions. Thus, for spectral multiplexing we block fewer nodes to have a smaller jammed region. One further difference between these two types of strategies is that spectral multiplexing typically requires more buffer space for storing packets during multiplexing. Given the limited buffer space on the motes, we chose to adopt a lower data rate (1 packet every 10 seconds) than the rate in earlier experiments.

Synchronous Spectral Multiplexing

In the synchronous spectral multiplexing experiment, we used the same general layout as shown in Figure 5.7(a). After the network had run for 40 packet intervals, the jamming/interference process began. The jammed region consisted of nodes 52, 53, and 10, and these three nodes promptly switched to channel 2. After the jammed nodes evaded to the new channel, their former parents detected their disappearance, and became the boundary nodes. Nodes 41 and 45 were such examples because they used to be the parents for nodes 10 and 52, respectively. The boundary nodes then announced themselves on the new channel, and waited to be selected as parents by the nodes on channel 2. In this experiment, all three jammed nodes chose node 41 as their parent. Thus, node 41 started working on two channels, while node 45 went back to work on channel 1. Overall, node 41 had four child nodes, three on channel 2 (nodes 52, 53, and 10), and one on channel 1 (node 44). In this experiment, the slot duration was 6.1 seconds,

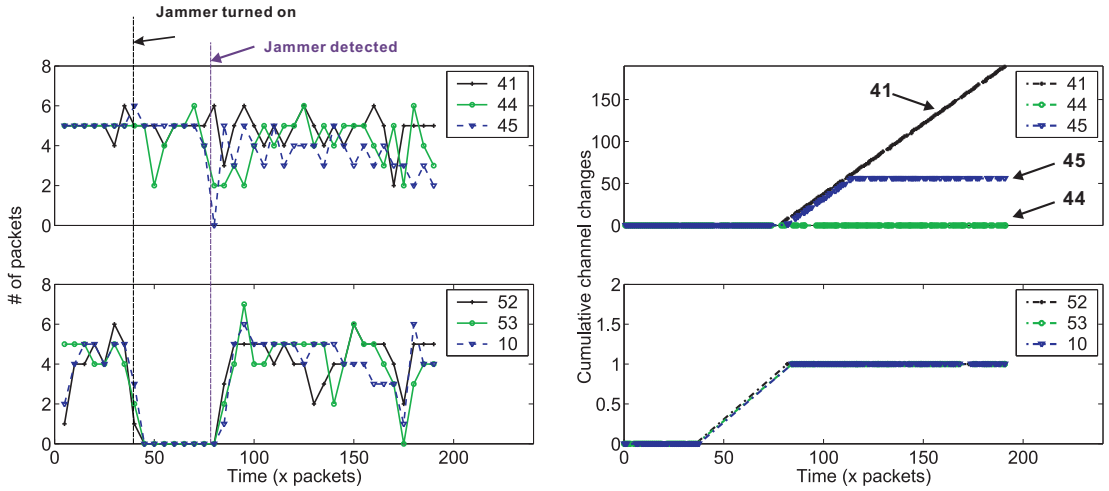


Figure 5.10: Statistics for the synchronous spectral multiplexing strategy: (a) number of packets delivered to the sink in a 50-second window vs. time, (b) total number of channel switches vs. time.

so that the number of packets buffered per channel was roughly 3 to 6 packets.

Figure 5.10(a) presents the time series of the number of packets delivered to the sink from six nodes in a 5-packet window. These six nodes include the three jammed nodes (nodes 52, 53, and 10), one boundary node (node 41), one potential boundary node (node 45), and one child of the boundary node that worked on channel 1 (node 44). The plot shows that the jammed nodes resumed their normal packet delivery performance. All other nodes were not affected much by the jammer. The plot shows that the jammed nodes resumed their normal packet delivery performance. All other nodes were not affected much by the jammer. The interval during which the jammed nodes had disrupted services was roughly 48 packet intervals, which is similar to the recovery time in the coordinated channel surfing strategy. Again, during the total recovery latency of 50 packet intervals, the boundary nodes waited for around 39 packet intervals before switching to the new channel to search for their children. After the boundary nodes found their children on the new channel, it only took them 11 packet intervals to start working on dual channels.

We report the total number of channel switches for these six nodes in Figure 5.10(b). These

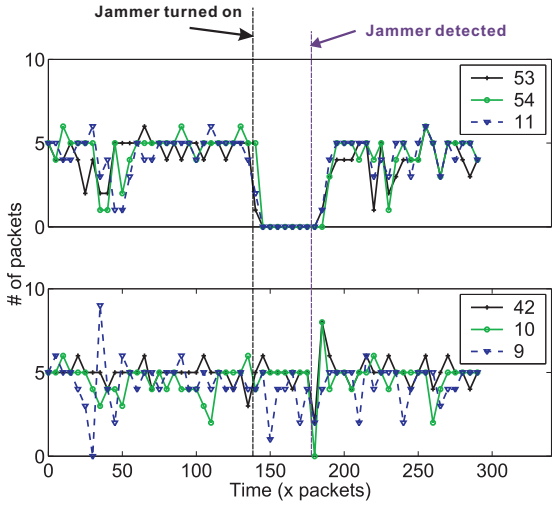


Figure 5.11: Packet delivery time series for asynchronous spectral multiplexing.

numbers agree with the discussion above regarding how different nodes responded to the emulated jamming in the experiment (e.g. node 41 continually switches channels). Interestingly, as noted earlier, node 45 started out as a dual-mode node and it initially switched channels frequently. However, after a short period of time, it was not selected as a dual-mode parent for any nodes on channel 2. It then returned to channel 1 and no longer switched channels.

Asynchronous Spectral Multiplexing

Our asynchronous experiment used a similar setup as the synchronous multiplexing experiment. The jammed region consisted of nodes 53, 54, and 11. After the jammed nodes evaded to channel 2, their former parents, i.e. nodes 42 and 10, also switched to channel 2, and announced their willingness to work on channel 2. Node 42 was chosen to be the parent by all three jammed nodes, and started flipping between both channels in a round-robin manner, while node 10 continued on channel 1. From then on, node 42 had three children (the three jammed nodes on channel 2).

Figure 5.11(a) presents the time series for the number of packets delivered to the sink from the above-mentioned nodes in a 5-packet window: the three jammed nodes (nodes 53, 54, and 11), one boundary node (node 42), and one potential boundary node (node 10). It is clear from

this figure that the asynchronous multiplexing scheme can quickly recover the connections between the jammed nodes and the sink without affecting other nodes in the network. As in the case of synchronous multiplexing, this scheme also incurred roughly a 50-packet service disruption period for the jammed nodes. We also recorded the total number of channel switches for these six nodes and observed trends analogous to those reported for synchronous multiplexing.

5.5.4 Channel Surfing Discussion

As we noted, we designed the channel surfing strategies so that the protocols do not rely on extra beacons or messages. Therefore, there is no additional overhead needed if the network is in a normal state.

Further, though the results presented above have shown that the four channel surfing strategies fare comparably, we emphasize that it is better to evaluate these strategies according to the network and interference scenarios for which they are most appropriate. The autonomous channel switching strategy relies on each individual node to detect whether its lost neighbors have been jammed and have escaped to the next channel. It does not introduce any additional protocol overhead, which can help the protocol design of a sensor network simple and clean. Its less proactive nature, however, leads to a longer delay in switching the entire network. Broadcast-assist Channel Switching, which requires all network nodes to switch their channel, is most suitable for cases where a large region is jammed, and the jamming occurs on a longer time scale (e.g. from a long-duration unintentional interference source). Spectrum multiplexing, however, is more effective for transient jamming where a few nodes are affected for a short duration. Here, we should determine whether to adopt synchronous or asynchronous spectral multiplexing based on the underlying traffic model. For instance, synchronous spectral multiplexing is more suitable for regular traffic patterns, while asynchronous spectral multiplexing can better cope with irregular (e.g. bursty) traffic.

5.5.5 Channel Following Jammers

We were also interested in more challenging interference scenarios, such as the scenario in which the jammer follows the network as it channel surfs. We conducted experiments with this

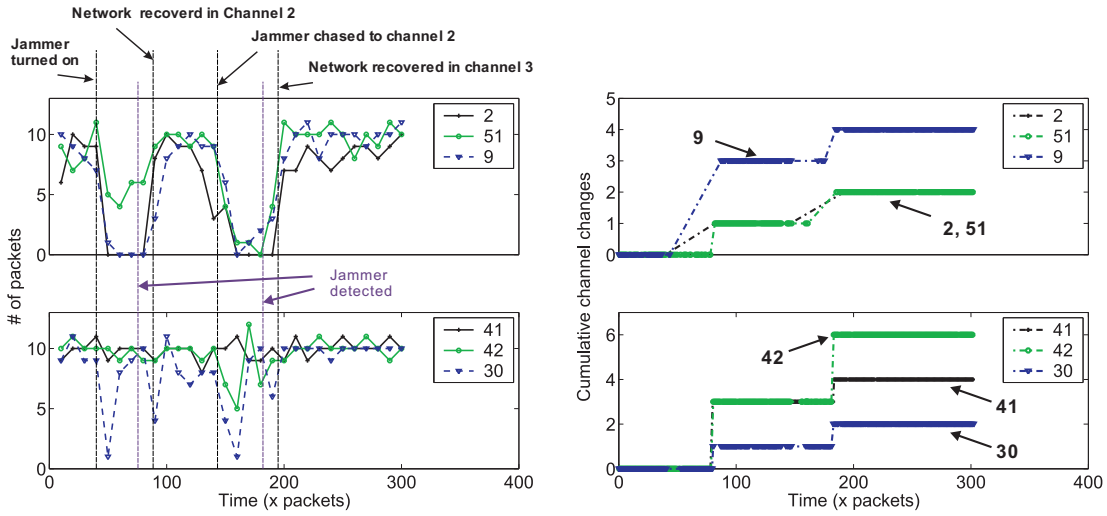


Figure 5.12: (a) Packet delivery time series for the broadcast-assist strategy when the jammer follows the network's channel surfing. (b) Time series illustrating the amount of times a node has changed its channel during the network's operation.

new jammer model, and found that all the four proposed schemes could restore network connectivity in such cases. Due to space limits, we do not provide results for all four schemes here, but rather we show the experimental results for the broadcast-assist channel switch scheme.

The network setup was the same as in Figure 5.7(a). We started the network on channel 1, and then introduced the jammer approximately at the 40th packet into the experiment, as depicted in the time series in Figure 5.12 (a). Shortly thereafter, the network adapted, with all nodes switching to the second channel after an overall latency of approximately 47 packet intervals. We allowed the network to run in the new channel for 50 packet intervals to find the new channel the network moved to, and thus the jammer switched to channel 2 at the 140th packet interval. The network again adapted to the interference, switching to the third channel after a total latency of roughly 51 packet intervals. Examining the packet delivery time series for node 51 illustrates an interesting phenomena regarding the effectiveness of a jammer: when the jammer was on channel 1 it was not entirely effective in disrupting the operation of node 51 while when the jammer was on channel 2, it was more effective at disrupting the communications from node 51. We believe this is due to the irregularity of the Mica2 radio. In addition to packet delivery traces, we recorded the amount of times different nodes switched

channels during the experiment, as illustrated in Figure 5.12 (b). This curve points to the fact that the protocol does not require an excessive amount of channel switching, typically requiring either one or three channel change attempts prior to settling in on the new channel. We draw the reader's attention to the channel change trace for node 41 and node 42. During the first channel change, these nodes switched back to the original channel to broadcast the change channel command, thus requiring a total of 3 channel changes. However, after the jammer follows them to the new channel, both the radio dynamics and the underlying network topology have changed, and in this case only node 42 was involved in switching back to channel 2 to announce the change channel command.

5.6 Summary and Conclusion

It is foreseeable that, as wireless sensor networks become increasingly deployed, they will be subjected to increased levels of radio interference. Such interference may be intentional, such as might arise from a jammer, or may be incidental, as may occur due to the presence of other wireless networks. In either case, most commercial wireless sensor networks will be susceptible to radio interference and, as a result, the ability of the sensor system to feed data to monitoring applications may be undermined. It is therefore critical to develop methods that can make sensor networks coexist with each other and even survive external interference. These defense mechanisms, however, must be distributed, easy to scale, and have low false positives. We have tackled this challenge in this chapter by presenting a family of channel surfing strategies that may be used to restore connectivity in the presence of radio interference. We presented two families of channel surfing strategies: the first, which we refer to as channel switching, consists of techniques whereby the entire sensor network changes its operating frequency; the second family, which we refer to as spectral multiplexing, aims to change the operating frequency in a neighborhood local to the interference, with boundary nodes acting as a radio-bridge across different channels. We implemented our strategies on a testbed of Mica2 motes, and have reported their performance for several interference scenarios. We found that our broadcast-assist strategy, as well as our multiplexing schemes, can effectively repair the network with short latency.

Chapter 6

Background and Related Work

6.1 History of jamming

Radio interference attacks have been studied in the electronic warfare community for some time, particularly in the context of RADAR systems. In this chapter, we will briefly review the history of jamming by looking at how radio interference has historically been considered in the radar and communication fields. Throughout the discussion, we will place our work in the context of other work that has been done in interference.

6.1.1 Intentional jammers

As early as World War II, a variety of radio jammers were used to disturb the operation of an enemy's radio devices (both communication and RADAR devices). RADAR was widely used during the war to detect the intrusion of enemies and to guide missiles and aircrafts in battle. Since such capabilities represent a significant advantage in a conflict, it was paramount that an enemy develop effective methods to interfere with the operation of a RADAR. There are two types of radar jamming: mechanical jamming and electrical jamming.

Mechanical jamming uses special metal objects to create false targets/images in the RADAR output. For example, chaff, as show in Figure 6.1 (a), is made of aluminium, metalized glass fiber, or pieces of plastic, with varying lengths so as to effectively reflect and scatter a broad range of RF frequencies. The delivery of chaff involves deploying an explosive device that, when exploded, would spread chaff in a cloud, thereby creating a cluster of secondary targets on radar screens, as depicted in Figure 6.1 (b). Another well known type of mechanical radio jammer is the class of decoys [5], which are flying objects that seek to deceive a radar into believing that they are actually aircraft, causing the radar to attack it or change the tracking focus.

Another type of RADAR jamming technique, electrical jamming, involves the radiation of a high-power signal to overshadow the real RADAR signal. As a result, the RADAR cannot

(a) (b)

Figure 6.1: (a) An Lancaster chaff (the crescent-shaped white cloud on the left of the picture) from within the accompanying bomber stream. (b) The effect of chaff on the display of a German Giant Würzburg radar.

correctly identify intruders. There are many variations of electrical jamming: spot jamming whereby a jammer focuses all of its power on a single frequency; sweep jamming, whereby a jammer's full power is shifted from one frequency to another; and barrage jamming, whereby a single jammer jams multiple frequencies at once.

Beyond interfering with the operation of RADAR systems, jamming methods have been employed to interfere with wartime communications. Jamming foreign radio broadcast stations was frequently employed during and after World War II to prevent or deter citizens from listening to broadcasts from enemy countries. During the Cold War, Soviet jamming of Western broadcasters led to a "power race" in which broadcasters and jammers alike repeatedly increased their transmission power capabilities, antenna directivity was improved in order to gain a directional advantage, and additional frequencies were used to make it difficult for jammers to interfere with communications.

In the modern civilian world, there are many commercial jammer products that can be purchased. Some of these can be used for good purposes. For example, cell phone jamming units can be purchased and installed to block cell phone reception. It is desirable to install those cell phone units in movie theaters or in classrooms. On the other hand, some commercial jammers are illegal, for instance, police speed gun jammers, which can prevent speed guns from working properly, are available but are illegal.

6.1.2 Unintentional jammers

As wireless networks become increasingly pervasive, it is very likely that the radio environment will no longer be favorable. There are several reasons for this gloomy forecast. First, as an increasing number of wireless devices (sensors, mesh networks, or otherwise) are deployed that use "open spectrum" bands (e.g. the 900MHz, 2.4GHz and 5GHz bands), it will be inevitable that there will be problems of spectrum coexistence. This is the well-known near-far problem of communications, which manifests itself in dense urban environments where cordless phones (sharing the same spectrum) suffer degraded performance when many devices operate simultaneously within a small distance of each other. Such interference problems can be projected for sensor networks/WiFi networks as more sensor devices/WiFi devices are deployed in these bands. Another reason stems from the fact that most different types of networks will share the same open spectrum. For instance, 2.4GHz spectrum has become increasingly crowded, as cordless phones, 802.11b/g devices and Bluetooth devices all operate in this band [17, 18, 55]. Even the microwave oven has energy leakage near 2.5GHz.

6.2 Related work

Radio interference attacks are a serious threat to the operation of a wireless network, regardless of the type of wireless network. Much work has been dedicated to understanding the different threat models that may be employed by adversaries, the methods that are needed to diagnose these threats, and the countermeasures that may be employed to defend against jamming attacks. In this section, we will overview the literature related to radio interference as well as other similar efforts devoted in the area of wireless networks.

Radio Interference and Jamming Attacks

Malicious Jamming Attacks at the PHY layer: Generally speaking, the objective behind jamming a communication system involves trying to disrupt the ability of a receiver to receive information that is intended to be transmitted to that receiver. The malicious jamming attacks in the PHY layer involve interfering with successful PHY-layer decoding of legitimate packets at the receiver side. In particular, jamming a communication receiver at the PHY layer can involve exploiting many different levels of the receiving operations. For example, one might

seek to affect the receiver front end by trying to saturate the dynamic range of the receiver, or one may strive to affect gain control processes so that the receiver is not able maintain power levels [51]. For example, a quick pulse jamming signal can cause the gain of a receiver to rapidly lower, affecting how the receiver will calibrate amplitude of the next received signal. Should an incoming signal arrive before the gain has recalibrated, this signal will have an increased chance of being improperly decoded.

Malicious Jamming Attacks in the MAC layer: Jamming attacks in the MAC layer involve disrupting information transfer by exploiting knowledge of the MAC protocols. Preventing such information transferral can be accomplished by either targeting the transmission of the signal (i.e. jam the sender), or by targeting the reception of the signal (i.e. jam the receiver). As noted earlier in this thesis, targeting the transmitter typically involves exploiting the scheduling of transmissions at the transmitter. For example, by using the fact that the medium access layer might perform carrier sensing in order to prevent collisions, can also be exploited to prevent the sender from sending by merely raising the ambient energy so that the sender thinks the channel is always busy. On the other hand, jamming can also target the receiving functions that depend heavily on error detection. For example, in this thesis, the reactive jammer can effectively disrupt communication because it causes CRC checks to fail at the receiver side, which naturally prevents packet processing from continuing.

Additional jamming strategies were studied by Law et al. [40], where jammers are targeted at jamming S-MAC [80], LMAC [13] and B-MAC [58] in an energy-efficient manner by carefully emitting a jamming signal based on the observed packet inter-arrival times so as to interfere with the regular packets sent. The efficiency of these methods was quantified in terms of the amount of resources needed to conduct an attack. Further work on jamming has studied MAC-layer jamming attacks on reservation-based medium access control schemes [62].

Jamming Detection

The issue of jamming detection was briefly studied by Wood and Stankovic in [77] in the context of sensor networks. This study posed the issue of jamming detection in the loose context of the utility of the communication channel. The idea is to detect non-transient jamming by examining whether the utility of the channel is below a certain predefined threshold. The

paper presented several factors that might affect the channel's utility, such as repeated inability to access wireless channel, bad framing, checksum failures, protocol violations (e.g., missing ACKs), excessive received signal level, low signal-to-noise ratio, or repeated collisions. Those factors, however, were not studied in detail, nor in a real system implementation. Instead, the paper primarily focused on providing jammed-area mapping services. In this thesis, we have investigated many single measurements that might be used to detect jamming attacks. Our experiments show that no single measurement is a *sufficient* statistic for basing jamming detection decisions upon. To address the jamming detection problem, we have explored the inconsistencies that might arise from naively employing decision processes built upon single factors. In particular, our detection algorithms may be viewed as a complement to the work of Wood and Stankovic and, when integrated with their mapping algorithm, can lead to enhanced mapping services.

Countermeasures for Coping with Radio Interference

Spread Spectrum: The traditional literature on coping with jamming primarily focuses on the design of physical layer technologies, such as spread spectrum [61,69,70]. Spread spectrum techniques involve transmitting information over a broader frequency band than is otherwise needed. The purpose of using extra bandwidth for transmission is to provide resistance to RF jamming/interferenace. There are typically two ways to achieve spreading of the spectrum needed to communicate a message: one is frequency hopping, which involves switching the carrier frequency for a narrow-band information-carrying signal across a broad spectrum; the other is direct sequence spread spectrum, which involves directly mapping information bits into a broad spectrum signal through application of spreading sequences.

A common misconception is that spread spectrum techniques are very resistant to jamming. This is, actually, not entirely true as spread spectrum receivers depend on a carrier synchronization process that might, itself, be the target of jamming. A clever attack on synchronization circuitry can cause the entire signal to be decoded incorrectly.

Even so, it should be realized that the physical layer technologies needed to reliably resist jamming have not found widespread deployment in commodity wireless devices, such as wireless LANs and sensor networks. Instead, most commercial platforms use simpler radios

with carrier sensing. Consequently, any form of radio interference that sufficiently elevates the energy in a channel can prevent a sensor radio from accessing the channel, effectively disrupting communications. We note, though, that the PHY in platforms like the MicaZ or 802.11 do employ a minimal amount of spreading in order to have multipath resistance. In this thesis, our work takes the viewpoint that rather than replace existing systems with more complicated radio platforms, it is instead desirable to understand the modes of attack that may be launched against existing platforms, and be able to detect them. Following detection, appropriate countermeasures may be employed.

Low Density Parity Check(LDPC): Countermeasures for coping with jammed regions in wireless networks were studied in [11, 45, 52]. In [52], the authors investigated the use of various coding schemes to improve the reliability of communication in the presence of radio interference. The use of low density parity check (LDPC) codes is proposed as the suitable technique to cope with jamming. Together with LDPC, the techniques of cryptographic interleaving are used to hide the structure of sub-data block permutation. As a result, the jammer cannot analyze the interleaving pattern, and "smartly" place its jamming energy across the transmission of packets to cause packet errors at a minimum jamming energy cost. Further, an anti-jamming technique is proposed for 802.11b that involves the use of Reed-Solomon codes.

Spatial Retreats: Spatial retreats involves mobile sensor nodes physically moving away from the interference source. Preliminary algorithms for various wireless networks are presented in Chapter 5.2. The issues of network partitioning that might arise as a result of mobile jammers are studied in [45]. To achieve robustness to mobile jammers, an additional reconstruction phase that follows the escape phase is proposed to reestablish connections after the mobile jammer moves away.

In the reconstruction phase, network nodes will move back to the coverage holes within which the mobile jammers is no longer present. The direction of the movement, and the destination position of the movement are controlled by the virtual forces arising from a system potential function that is targeted at repairing the network coverage. As the reconstruction phase is a continuous process, the network nodes will continuously repair the network topology regardless of the jammer movement.

Wormhole-based Anti-Jamming Techniques: In [11], wormhole-based anti-jamming techniques are studied analytically. Under a jamming attack, the network tries to build a wormhole between nodes in jammed regions and nodes outside of jammed regions, through which important messages can be delivered out of the jammed regions. Three types of techniques are studied to create probabilistic wormholes. First one involves using wired pairs of sensors; the second is based on frequency hopping pairs; and the third relies on uncoordinated channel hopping. The proposed approaches try to ensure the successful delivery of at least one alarm message for each event, instead of resuming continuous network connectivity. Our work builds on these efforts by providing system validation.

Other Related work

Mapping Jammed Areas: As noted earlier there are efforts devoted to mapping out regions of the sensor network that are jammed. By having a map of jammed-areas, network services can use this knowledge to influence routing, power management, and higher-layer planning. A protocol for mapping out the jammed regions of a sensor network was presented in [77]. In this paper, jamming detection is performed by monitoring channel utilization. Once the sensors observe that their channel utility is below a preset threshold, they conclude that they are jammed. Following detection, the jammed nodes bypass their MAC-layer temporarily and broadcast JAMMED messages, announcing the fact that they are jammed. These JAMMED messages will not be able to be received by other jammed neighbors. However, those neighbors on the boundary of the jammed region, but are not themselves jammed themselves, will be able to hear the JAMMED messages, though potentially at a higher error rate. Once non-jammed sensors receive JAMMED messages, they initiate the mapping procedure. These non-jammed nodes exchange and merge information describing which nodes they have witnessed as jammed, where those jammed sensors are located, along with neighbor information. By continuing the exchange of information regarding witnessed jammed nodes, the network will eventually be able to map out the boundary of a jammed area.

Increasing the Network Throughput by Multiple Channels: The use of multiple channels has been proposed as a means to enhance the throughput and performance of wireless mesh networks, e.g. [63,67,72], and cellular networks. In this area, the radio devices are resource-rich

compared to sensors, and often have multiple-radio interfaces. The challenge here is assigning the channels and/or time slots across multiple nodes to avoid collisions and congestion. These works are primarily intended to enhance performance in normal conditions, and do not attempt to cope with unexpected jamming/interference, as is described in this dissertation. Generally, channel allocation for these scenarios is achieved with the aid of a centralized entity or via a common control channel. In our work, we have not resorted to centralized entities or control channels in order to achieve jamming resistance.

Greedy User Behavior: Although not precisely a jamming attack, one may exploit the MAC layer to achieve increased network resources [8, 38]. The issue of detecting non-MAC compliancy was recently studied in [66]. This work showed that a greedy user can increase his share of bandwidth by sightly modifying the driver of his/her network adapter. The greedy user may try to corrupt the RTS and CTS of other users to prevent packet transmission, or may corrupt ACKs to cause the ACK contention window to increase, leading to larger backoff. They proposed DOMINO, a system for detection of such greedy behavior in the MAC layer of IEEE 802.11 public networks.

Other DOS Attacks in Wireless Networks: In 802.11 networks, individual devices claim their identity by providing a unique MAC address, and no authentication mechanisms are natively deployed to verify the self-report identity. Thus, it is very easy for an adversary to launch identity spoofing attacks, which could lead to many vulnerabilities that involve deauthentication, disassociation, and evil twin access points (AP) [8]. Since deauthentication request messages are not authenticated, any adversary can "spoof" a client or the access point to deauthenticate from one another. As a result, the client and the AP will deny any further packets unless authentication is reestablished. Similarly, an attacker can "help" the client to disassociate itself from the AP, which will stop the AP from forwarding packets to and from the client. Additionally, an adversary can pretend to be a legitimate access point by broadcasting the same MAC address and ssid with a stronger signal. By default, a client will automatically associate with the rogue access point, which has a stronger signal. Consequently, the adversary can compromise communication by performing various malicious operations, including directing fake traffic to the associated client, or dropping the requests made by the client.

To prevent a radio device from altering its network identifiers to that of another network

device, identity authentication methods are necessary at the lower layers. Several conventional network authentication methods can be used in 802.11 networks. An alternative form of authentication, which does not rely on the explicit use of authentication keys to identify entities but instead relies on forge-resistant relationships associated with packets, is studied in [43]. Relationship based spoofing detection techniques involve, for example, the use of monotonic relationships in the sequence number fields, or the enforcement of statistical characteristics of legitimate traffic to detect multiple devices using the same network identity.

Chapter 7

Concluding Remarks

7.1 Thesis Summary

Wireless networks are being deployed in a variety of forms, ranging from ad hoc networks to wireless LANs to sensor networks. The shared nature of the wireless medium makes wireless networks particularly susceptible to a variety of threats, intentional or otherwise, that can undermine the ability to communicate. In particular, it is possible for other devices to interfere with communications by actively broadcasting– either as unintentional interferers sharing the same bandwidth, or as intentional adversaries seeking to disrupt communications. Therefore, understanding the nature of jamming attacks and developing methods that can make commodity wireless devices survive jamming attacks are critical to assuring the operation of wireless networks.

We have provided a brief survey on the problem of defending against radio interference for three broad wireless communication scenarios: two-party radio communication, an infrastructured wireless network, and an ad hoc wireless network. We have presented evasion strategies that may be employed by wireless devices to evade radio interference for various wireless network. The first strategy, channel surfing, is a form of spectral evasion that involves legitimate wireless devices changing the channel that they are operating on. The second strategy, spatial retreats, is a form of spatial evasion whereby legitimate mobile devices move away from the locality of the jammer.

The rest of the thesis has been focused on defending ad hoc network from radio interference. In particular, the rest of the thesis has sought to focus on both the attack and defense side of *jamming* in wireless networks. Although there are many scenarios where interference will occur, we have focused our attack/interference discussion on four different types of jamming, which cover both malicious jamming attacks and unintentional radio interference. For these jamming scenarios, devices either intentionally bypass the traditional MAC layer carrier sensing, or unintentionally ignore MAC layer issues as they employ a different radio technology.

We have studied the effectiveness of our four jammer strategies by constructing prototypes using the MICA2 Mote platform and have measured how each of the jammers fared in terms of their effect on the packet send ratio and packet delivery ratio. Our experimental results show that all four jammers are effective in disturbing the network communications.

We then studied the issue of detecting the presence of jamming attacks, and examined the ability of different measurement statistics to classify the presence of a jammer. We showed that by using signal strength, carrier sensing time, or the packet delivery ratio individually, one is not able to definitively conclude the presence of a jammer. Therefore, to improve detection, we introduced the notion of *multimodal* detection methods, where the packet delivery ratio is used to classify a radio link as having poor utility, and then a consistency check is performed to classify whether poor link quality is due to jamming. We introduced two enhanced detection algorithms: one employing signal strength as a consistency check, and one employing location information as a consistency check. We evaluated the effectiveness of each scheme through empirical experiments and showed that each of the four jammer models we introduced can be reliably classified using our consistency checking schemes.

Following detection, it is desirable that the network can repair itself, and thus proper defense mechanisms are necessary to make wireless devices survive external jamming attacks. These defense mechanisms, however, must be distributed, easy to scale, and have low false positives. We have tackled this challenge in this thesis by presenting a family of channel surfing strategies that may be used to restore connectivity in the presence of radio interference. We presented two families of channel surfing strategies: the first, which we refer to as coordinated channel switching, consists of techniques whereby the entire sensor network changes its operating frequency; the second family, which we refer to as spectral multiplexing, aims to change the operating frequency in a neighborhood local to the interference, with boundary nodes acting as a radio-bridge across different channels. We implemented our strategies on a testbed of Mica2 motes, and have reported their performance for several interference scenarios. We found that our broadcast-assist strategy, as well as our multiplexing schemes, can effectively repair the network with short latency. Our study serves as an initial systems effort towards building interference-resistant sensor networks and future investigations will involve examining more advanced jamming scenarios.

7.2 On-going and Future Work

Building large-scale, interference-resistant networks will require a long-term effort by the entire community. This thesis is intended to serve as a starting point towards this goal, and there are several interesting possibilities for future work.

One important direction that warrants investigation involves studying channel surfing strategies in the presence of more advanced jamming/interference scenarios, such as multiple jammers conducting channel chasing. Coping with channel chasing jammers place stringent requirements on the response time. The delay between the moment that the jammers start to blast on the channel to the moment that the network detects the jamming attacks and thus resumes its connectivity in the new channel should out-compete the speed by which the jammers chase the network.

7.2.1 Competition strategies

An alternative to performing channel surfing, where the wireless devices try to evade the jammer in the frequency domain, is to have them attempt to compete against the jammer. In this case, the objective is for the wireless devices to improve the reliability of the reception of their packets. This requires, if a node detects it is jammed, that it will ignore the fact that it is jammed, and transmit its packet anyway. In order for nodes that are jammed, as well as nodes near jammed regions to compete against the jammer they should adjust the error correcting coding and the power of their communications at the lower layers. By employing a stronger error correcting code with a lower information rate, we reduce the overall throughput but increase the likelihood of packets successfully being decoded. We may also increase the transmission power employed by legitimate radio devices in order to allow reception to operate at higher SNRs.

Power Control: This strategy involves a return to looking at the problem using traditional communications theory, as well as methods employed by the electronic warfare community. By applying the traditional communication theory presented in Section 3.1, we come to the conclusion that, in the case where the noise power is much smaller than the jamming power,

the channel capacity is approximately

$$C = B \log_2(1 + \frac{P_s}{P_j}) \tag{7.1}$$

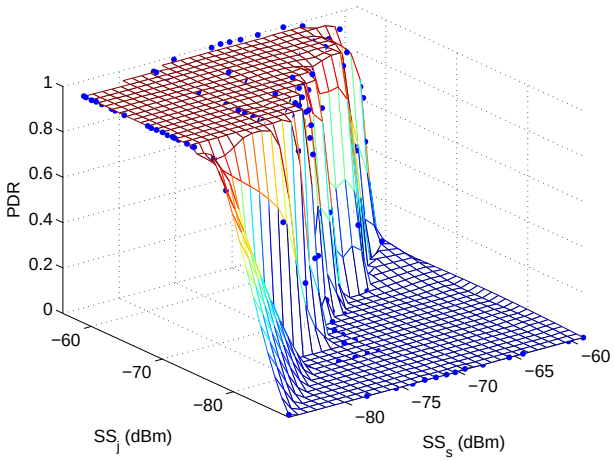
where P_s is the average transmission power of signal S, P_j is the average transmission power of jamming signal J. Thus, for a given jamming power, raising the transmission power of the sender A can increase the channel capacity.

To validate the feasibility of defending wireless networks against radio interference by employing power adjustment, we performed a preliminary experimental study on a two party scenario, where we deployed the sender S, the receiver R, and the jammer J at fixed location on MICA2 mote platform. Each mote has a 433 MHz ChipCon CC1000 RF transceiver and used TinyOS 1.1.16 as the operating system. To build the jammer J, we disabled the back off operations to bypass the MAC protocol. Rather than employing a constantly transmitting jammer, we employed the *reactive jammer* from Section 3.2, where jammer J listens to the channel and, upon detection of a preamble, it immediately blasts on the channel by sending a random 2-byte jammer packet.

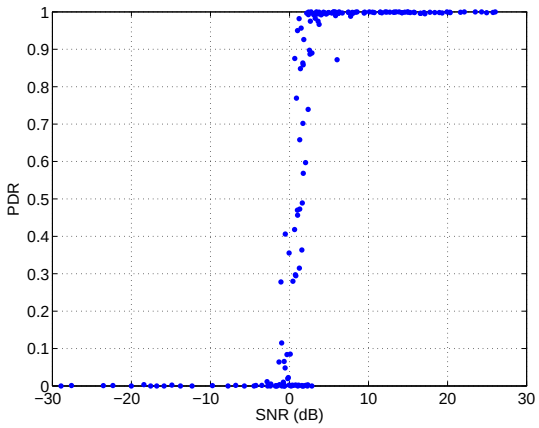
Translating channel capacity into a real system metric, we used Packet Delivery Ratio (PDR) to indicate the quality of a link as the source increases its power. We measured the PDR at the receiver R by calculating the ratio of the number of packets that pass the CRC check with respect to the total number of sent packets. The total number of sent packets can be obtained through tracking the sequence number of each packet. If no packets are received, the PDR is defined to be 0.

We placed S, R, and J at the coordinates (10,2) (10,11), and (7,4) respectively. We gradually increased the transmission power levels of both the jammer (P_j) and the sender (P_s) from $-20dBm$ to $10dBm$, which led to receive levels ranging from $-80dBm$ to $-55dBm$. For a given tuple (P_j, P_s), we measured the PDR over 2000 packet trials, and plot the result in Figure 7.1.

Figure 7.1 (a) demonstrates a sharp cliff phenomenon: for any jamming power, as we gradually increase the sender's transmission power above curtain threshold value P_t, the PDR will go back to 100%. The threshold value P_t is a function of the jamming power. The relationship between P_t and the jamming power can be better observed in Figure 2.2 (b) where we plot



(a)



(b)

Figure 7.1: (a) Packet delivery rate as a function of the transmission power level of the sender S and the jamming power, (b)Packet delivery rate as a function of signal-to-jamming ratio.

the PDR measurements as a function of $\gamma_{S/J}$, the ratio of the received signal power over the jamming power. In this figure, we observe that the PDR jumps approximately at $\gamma_0 = 2dB$. We note that in our experiment, we did not subtract the ambient noise level from the received signal and jamming signal measurement.

Our PDR results of the two-party scenario validate that, by increasing the transmission power of the sender S, S can overshadow the jammer J, and thus successfully deliver packets to the receiver. However, the key difference between two party scenario and a general wireless network deployment suggests it is not trivial to apply power adjustment to various wireless networks for anti-jamming. In particular, since we are operating in a multihop network scenario, it should be realized that using increased power levels introduces new problems as radio devices

will have a larger radio coverage pattern, thereby increasing the likelihood of collisions and unintentional interference with other legitimate radio devices.

Error Correction Codes: Competition strategies are at an early stage of investigation. Although we have examined power-adaptation mechanisms in two party scenarios, one additional direction to explore is adjusting the error correction coding at the link layer to improve the reliability of the packet reception. The goal would be to design a protocol that each network device can following to dynamically alter its coding rule based on its local radio condition. We note that there are many practical issues that need to be solved. For example, the sender should be able to adjust its error correction coding so that its potential receiver can successfully receive and decode its packets. The sender, however, might not have the knowledge of the radio condition at the receiver side, which makes error control adjustment hard to conduct.

Generally, just as channel surfing was studied thoroughly through a testbed deployment, competition strategies require thorough systems-level investigations, in order to assess their value.

7.2.2 Timing channels

We have presented several defense strategies that are effective to repair the network connectivity in the presence of unintentional interferers or somewhat malicious jammers. In some cases, the proposed defending strategies, e.g. evasion strategies whereby the wireless devices evade from the radio interference sources in various ways, is not an option. For instance, if the radio interference could have spanned the entire frequency band that the wireless devices are able to operate on, then there is no interference-free channel left for the device to hop to. It could be that the power level of the radio interference is too high for the wireless devices to compete due to the limited power level the devices are capable of transmitting. In this section, we examine an alternative method to convey information in the presence of powerful radio interference. The goal here is not to increase the reliability of packet transmissions, but to construct a low-bit rate channel, though which important information can be conveyed.

Let us re-examine the Alice and Bob conversation example mentioned earlier, where an adversary Mr. X interrupts Alice and Bob by continuously telling inane stories. Suppose for some reason, Alice and Bob cannot excuse themselves from the conversation to meet up with

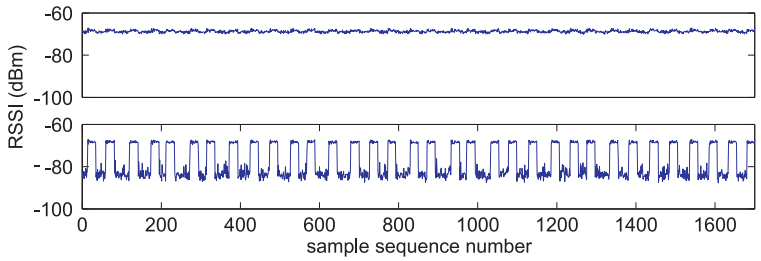


Figure 7.2: RSSI readings as a function of time in different jamming scenarios. The upper plot corresponds to a high power jammer scenario, and the lower plot corresponds to a jammer which wishes that its jamming signal is hard to detect, i.e. low probability of detection.

each other later in a different location. In this situation, Alice can, however, figure out what Bob wants to say by reading his lip movements, vice versa. Applying this idea to the context of wireless networks, wireless nodes can communicate with their neighbors via "lip movements", the presence of packets. Although the packets cannot be delivered without bit errors, the receiver can detect the presence of packets. The objective here is not for the wireless devices to improve the reliability of the reception of their packets, but to cleverly convey information by exploring contextual information, e.g. the fact that a packet has been received but it failed CRC checking.

Interference Model: The radio interference models we focus on in this discussion includes constant jammers and reactive jammers, as described in Chapter 3. A constant jammer continuously emits random bits or a random radio signal into the channel, whereby either prevents nodes from sending out packets or corrupts the packets on the fly. A reactive jammer starts transmitting a radio signal as soon as it senses the arrival of packets. Now we examine the likelihood for a receiver to detect that its neighbor has tried to transmit a packet. In the reactive jammer scenarios, the jammer stays quiet when the channel is idle, but starts transmitting a radio signal as soon as it senses activity on the channel. In parallel, network nodes will start to receive the packet as soon as they decode the preamble. However, the packets, which are garbled by the jammer, will be dropped at the lower layer as a result of the CRC checking failure. Being able to identify when the packets start suggests that we can record the packet arrival time sequence and convey information by modulating the inter-arrival time between two consecutive packets.

A constant jammer continuously interferes with network communication, which prevents the network nodes from recording the packet arrival time by means of identifying preambles of the packets. Alternatively, we can measure the packet arrival time by examining the ambient noise level. In order to understand the effect that a jammer would have on the received noise levels, we performed an experiment with Berkeley motes. Our experiments involved a single mote A merely measuring noise levels without transmitting any packet, a mote B sending packets periodically, and a constant jammer X that constantly transmits random bits into the air. We vary the jamming power of X while setting B's transmission power to minimum. We present time series data for two different jamming power settings in Figure 7.2. From the upper plot of this figure, it is immediately apparent that, if the jammer is a high-powered source of interference, then it is very unlikely to detect the presence of packets using a low cost radio receiver. However, in most cases, the objective of a malicious jammer is to interfere with the legitimate network communication with low energy consumption, or with low probability of being able to be localized itself. Thus, a minimum jamming power is often used to interfere with wireless network communication, which corresponds to the lower plot in Figure 7.2. The measured noise level time series with the jammer exhibit period variation with each spike representing a packet being sent. This observations suggest that pattern recognition techniques can be employed to identify the packet arrival time.

Timing Channel Overview: Assume that the times of packet arrivals/failures can be measured by the receiver. We now examine how to code data into packet inter-arrival times.

Consider the two party setup where we have a sender S communicating to a single receiver R. The sender S will send a stream of k packets (over a time period $[0, T]$) to convey a symbol $s \in \mathcal{S}$, e.g $s \in \{0, 1\}$. In order to map packet arrival times to messages, we introduce a definition:

Definition 1 *We define a single-sender (synchronous) arrival code* $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_m\} \subset [0, T]^k$ *where* $m = |\mathcal{S}|$. *Each* $\mathbf{c}_j = (c_{1j}, c_{2j}, \cdots, c_{kj})$ *has the monotonicity property* $c_{ij} < c_{lj}$ *when* $i < l$.

The codeword $\mathbf{c}_j$ may be thought of as the arrival times associated with a sequence of k packets. Associated with $\mathcal{C}$ is its inter-arrival code $\Delta_{\mathcal{C}}$, which is composed of codewords

$\Delta_{\mathcal{C}} = \{\delta_1, \delta_2, \cdots, \delta_m\} \subset [0, T]^{k-1}$, where

$$\delta_j = (c_{2j} - c_{1j}, c_{3j} - c_{2j}, \cdots, c_{kj} - c_{k-1,j}).$$

Note that there is an implicit requirement, i.e. $\|\delta_j\|_1 < T$ because $\|\mathbf{c}_j\|_1 < T$.

For a single-user arrival code, we define the minimal difference distance (MDD) of $\mathcal{C}$ by

$$d^*_{\mathcal{C}} = \min_{j,l} \|\delta_j - \delta_l\|_2$$

where $\|\mathbf{v}\|_2$ denotes the standard Euclidean norm. We may think of the minimal difference distance of a code as being the closest its codewords get to each other. One objective, then, is to design codes with large MDD for a given k, m and T. This will allow us to correctively decode communications should errors occur (e.g. possible calibration drift/jitter). On the other hand, we want to design codes so that the number of symbols that can be sent out for a given duration T is maximized.

Let's look at a simple example, $k = 2$, $m = 2$, where we have $\mathcal{S} = \{0, 1\}$, and $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2\}$. Then, by the fact that we are considering a period of T time units to convey either 0 or 1, we must have that $\delta_{1j} = c_{2j} - c_{1j} < T$, where $j \in \{1, 2\}$. Our problem of designing a code with good distance properties amounts to selecting two constellation points $\delta_{1j} \in [0, T]$. If we select $\delta^{\epsilon}_{11} = \epsilon$ and $\delta^{\epsilon}_{12} = T - \epsilon$, then we have two constellation points with "guardbands" of duration $T - 2\epsilon$. This corresponds to the arrival code $\mathbf{C} = \{(\alpha, \alpha + \epsilon), (\beta, \beta + T - \epsilon)\}$, where α and β are arbitrary and chosen to satisfy the appropriate time constraints.

Give a time interval T, we are interested in how many symbols, or how many bits information can be transmitted. Assume T is an integral multiple of ϵ, where $T = n\epsilon$, then we can place n constellation points at $\{\epsilon, 2\epsilon, \cdots, n\epsilon\}$, each separated by ϵ. By doing so, a sequence of n binary digits can be transmitted during an interval T. Thus, the transmitter is able to communicate one of 2^n symbols whose size is n bits, where $k = 2$, $m = 2^n$. As an example, consider $n = 4$, the inter-arrival code $\Delta_{\mathcal{C}} = \{\delta_1, \delta_2, \delta_3, \delta_4\}$, where $\delta_i = i\epsilon, i \in [1, 4]$, and one of the symbols in $\mathcal{S} = \{00, 01, 10, 11\}$ can be sent. For a given interval T, ϵ is in inverse relationship with n, the amount of bits that can be transmitted. To increase the communication capacity, it is desirable to choose smaller values for ϵ. However, the minimum value of ϵ is limited by many factors. First of all, it has to be larger than the amount of time that is required to complete one

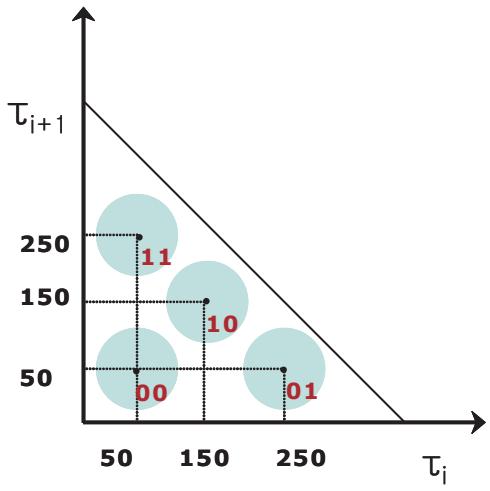


Figure 7.3: Triangular simplex for an inter-arrival code.

transmission. In MICA2 motes, it takes approximate $12ms$ to transmit a packet of $13bytes$. Additionally, uncertain latency, which is accumulated as the packet travels across various layers and buffers, can cause jitter of the packet arrival time. Those unwanted variations of the packet arrival times, which introduces error in the inter-arrival times, requires large MDD in order to accurately decode messages.

Experimental Validation: We choose to validate the idea of the timing channel by building a prototype where $k = 3$. Here, our arrival codeword $\mathbf{c}_j = (c_{1j}, c_{2j}, c_{3j})$ satisfies $c_{1j} < c_{2j} < c_{3j} < T$. The corresponding inter-arrival code $\delta_j = (\delta_{1j}, \delta_{2j})$ satisfies the property $\delta_{1j} + \delta_{2j} \leq T$, and $0 < \delta_{ij} < T$. In $\delta_1 \times \delta_2$ space, this maps out a simplex $\Delta = \{(\delta_1, \delta_2) | \delta_1 + \delta_2 < T, \delta_1 > 0, \delta_2 > 0\}$. In this case, the simplex Δ is a triangle.

In terms of our code design, we seek to put m circles of radius d within Δ so that these circles do not overlap, while having reasonably large $d^*_{\mathcal{C}}$, as depicted in Figure 7.3. To ensure that we have the ability to uniquely decode and to utilize the maximum bandwidth, we choose four circles centered at four constellation points: $\delta^\epsilon_1 = (\epsilon, \epsilon)$, $\delta^\epsilon_2 = (5\epsilon, \epsilon)$, $\delta^\epsilon_3 = (3\epsilon, 3\epsilon)$, and $\delta^\epsilon_4 = (\epsilon, 5\epsilon)$, where $\epsilon = 50$ms. The procedure of deciding what has been transmitted is to find the circle that encompasses the actual inter-arrival time measurement $\delta' = (\delta'_1, \delta'_2)$, e.g. to find $j \in [1, 4]$, which is associated with a symbol from $\mathcal{S}$, such that it satisfies$\|\delta_j - \delta'\|_2 < d$.

Before we could develop the proposed timing channel strategies on Berkeley Motes, we had to modify the existing TinyOS code to address a number of implementation related issues.

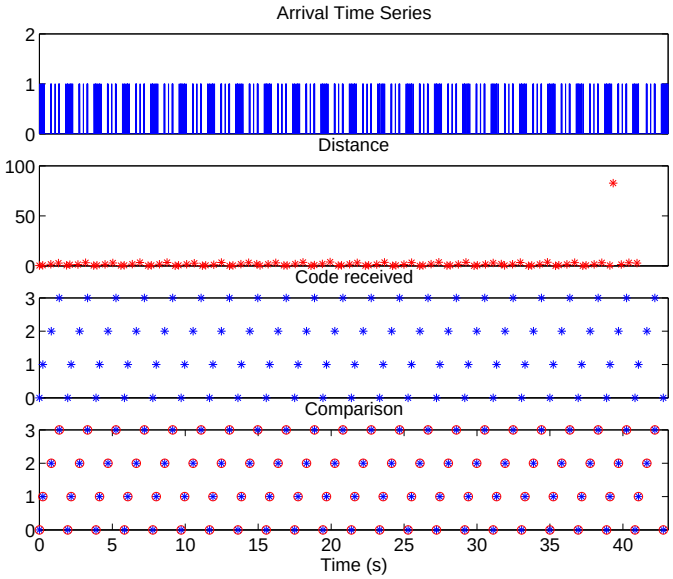


Figure 7.4: The results of timing channel experiment: the packet arrival series, the minimum Euclidean distance to the codewords, the decoded symbols, and the comparison between decoded symbols and the symbols sent.

The first challenge involves devising a deterministic packet transmission path, as this directly affects the decoding accuracies. The uncertain latency accumulated on the transmission path is the result of carrier sensing, and random back off at the MAC layer. Carrier sensing and random back off are useful mechanisms to ensure multiple access in a non-jamming scenario. In the jamming scenarios, those mechanisms are not necessary, and should be disabled to diminish jitter. After modifying the transmission path, we faced the challenge of recording the packet arrival time accurately. We have the incoming packets time stamped in the layer that is as close to the radio as possible to suppress jitter at the receiver side. Finally, an challenge arises due to the fact that a sequence of symbols are transmitted. It is necessary to identify the starting point of a symbol. In our implementation, we addressed this by having two consecutive symbols concatenate together with each separated by a interval ξ, which is much smaller than ϵ. For example, suppose we want to send symbols 01 after 00, we send three packets at time $(0, 50, 100)ms$ for symbol 00, then after pausing for $\xi = 25ms$, we send three packets at $(125, 375, 425)ms$ for symbol 01. In this way, receiving two packets with time separation of 25ms suggests the start of one symbol.

In our experiment, we have a sender S, a receiver R and a reactive jammer J. The sender

S cycles through the symbols $\{00, 01, 10, 11\}$ repetitively. The receiver R measures the packet arrival time and decodes the message. The results are depicted in Figure 7.4. The comparison sub-plot shows that the symbols decoded by the receiver R matches with the message the sender S sent modulated by packet interarrival time. Thus, it is feasible to convey information via packet arrival time in the presence of jamming attacks.

Multi-Senders: Constructing a timing channel in a multiple sender scenario is complicated by the fact that the packets coming from different senders are interleaved with each other. The challenge here is to somehow extract the packet inter-arrival time sequence modulated by different senders. Let's first define the problem formally. Consider the following set up: We have N senders $\{u_j\}_{j=1}^{N}$ communicating to a single receiver R. Each user will send a stream of k packets over a time period $[0, T]$ to convey a symbol $s \in \mathcal{S}$, e.g $s \in \{0, 1\}$.

Definition 2 *We define a multi-user (synchronous) arrival code*

$$\begin{aligned} \mathcal{C} &= \{\{\mathbf{c}_1^1, \mathbf{c}_2^1, \cdots, \mathbf{c}_m^1\}, && (7.2)\\ &\quad \{\mathbf{c}_1^2, \mathbf{c}_2^2, \cdots, \mathbf{c}_m^2\}, \cdots, && (7.3)\\ &\quad \{\mathbf{c}_1^N, \mathbf{c}_2^N, \cdots, \mathbf{c}_m^N\}\} && (7.4)\\ &\quad \subset [0, T]^K && (7.5) \end{aligned}$$

where $m = |\mathcal{S}|$. We note, that we have partition the code into sub-codes $\{\mathbf{c}_1^a, \mathbf{c}_2^a, \cdots, \mathbf{c}_m^a\}$ that correspond to each user u_a. As before, each $\mathbf{c}_j = (c_{1j}, c_{2j}, \cdots, c_{kj})$ has the monotonicity property $c_{ij} < c_{lj}$ when $i < l$.

In contrast to the single-user arrival code, we define the minimal difference distance (MDD) of a multi-user arrival code (MUD-MDD) $\mathcal{C}$ by

$$d_{\mathcal{C}}^* = \min_{j,l,a} \|\delta_j^a - \delta_l^a\|_2$$

where $\|\mathbf{v}\|_2$ denotes the standard Euclidean norm. Now the minimal difference reflects a design criteria across all of the users simultaneously. Clearly, with more users we introduce more constraints and it becomes harder to pack all of the codewords in the constrained region of $[0, T]^k$. Our objective, then, is to design codes with large MUD-MDD for a given k, m, N and T. One promising technique is to design the packet transmission sequence based on optical

orthogonal codes (OOC) [16, 68], which is used in optical communication for multiple access. The salient feature of OOC is that the cross-correlation between different code words is very small, while the auto-correlation is large. We may thus assign a non-overlapping set of OOC to different senders. Each sender will send out symbols by modulating the packet inter-arrival time based on its unique OOC code. The receiver can record the combination of packet arrival times from various senders. To decode the message, the receiver calculates the cross-correlation between the arrival time sequence and the OOC code. If the cross-correlation with the OOC code that is assigned to sender 1 as symbol s_i is larger than a threshold, then the sender 1 is likely to have sent out the symbol s_1. The challenge here is to design the OOC so that the false alarm is minimized.

Conveying information in the presence of radio interference via timing channel is at an early stage of investigation. Although we have validated the mechanism to transmit information by modulating the packet inter arrival time in the two party scenario in this dissertation, research is still needed to apply this techniques in multiple senders scenarios.

7.3 Final Remarks

Radio interference is a serious threat to the successful deployment of wireless networks. The problem of jamming is a fundamental challenge associated with radio technology, and it is not likely to go away. Much work has been devoted to jamming, especially in electronic warfare community in the context of RADAR systems. This dissertation has examinee the problem of radio interference in the context of commodity wireless networks. To cope with radio interference, we have taken the view point that instead of re-designing commodity radio devices to have sophisticated radio, we should explore the potential of existing radio devices and make them jamming resistant. Along this line, we have presented several defense strategies that can be employed on off-the-shelf wireless devices. In particular, we have evaluated the real impact of radio interference in field-tests, studied the effectiveness of radio interference detection strategies which involves consistency checks, and have examined the ability to cope with radio interference using various schemes, including adopting a different channel allocation, adjusting transmission power, and constructing a timing channel by modulating the inter-arrival times.

References

[1] Chipcon cc1000 datasheet. http://www.chipcon.com/files/CC1000_Data_Sheet_2_1.pdf.

[2] IEEE Std 802.11i/d3.0. Available at http://www.cs.umd.edu/ mhshin/doc/802.11/802.11i-D3.0.pdf.

[3] Mica2 homepage. http://www.xbow.com/Products/productdetails.aspx?sid=174.

[4] Tinyos homepage. http://webs.cs.berkeley.edu/tos/.

[5] D. Adamy. *EW 101 A First Course in Electronic Warfare*. Artech House Publishers, 2001.

[6] AusCERT. AA-2004.02 - denial of service vulnerability in IEEE 802.11 wireless devices. http://www.auscert.org.

[7] P. Bahl and V.N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of IEEE Infocom 2003*, pages 775–784, 2000.

[8] J. Bellardo and S. Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *Proceedings of the USENIX Security Symposium*, pages 15–28, 2003.

[9] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. *Advances in Cryptology - Crypto '96*, 1109:1–15, 1996. Lecture Notes in Computer Science.

[10] M. Bohge and W. Trappe. An authentication framework for hierarchical ad hoc sensor networks. In *Proc. of the 2003 ACM Workshop on Wireless Security*, pages 79–87, 2003.

[11] M. Cagalj, S. Capkun, and J.P. Hubaux. Wormhole-Based Anti-Jamming Techniques in Sensor Networks. *to appear in IEEE Transactions on Mobile Computing*, January 2007.

[12] S. Capkun and J.P. Hubaux. Secure positioning in sensor networks. Technical report EPFL/IC/200444, May 2004.

[13] S. Chatterjea, L. van Hoesel, and P. Havinga. Ai-lmac: An adaptive, information-centric and lightweight mac protocol for wireless sensor networks. In *the DEST International Workshop on Signal Processing for Sensor Networks*. IEEE Computer Society Press, 2004.

[14] C. Chiasserini and R. Rao. Co-existence mechanisms for interference mitigation between ieee, 2002.

[15] C. F. Chiasserini and R. R. Rao. Performance of ieee 802.11 wlans in a bluetooth environment.

[16] F.P.K. Chung, J.A. Salehi, and V.K. Wei. Optical orthogonal codes: design, analysis, and applications. *IEEE Transaction on Information Theory*, 35:595–604, 1989.

[17] M. Corporation. Wi-fi64 (802.11b) and bluetooth64: An examination of coexistence approaches.

[18] G. V. Dyck. Interference evaluation of bluetooth and ieee 802.1 lb systems.

[19] P. Enge and P. Misra. *Global Positioning System: Signals, Measurements and Performance*. Ganga-Jamuna Pr, 2001.

[20] F.H.P. Fitzek and M. Reisslein. MPEG-4 and H.263 video traces for network performance evaluation. *IEEE Network*, 15(6):40–54, November/December 2002.

[21] X. Fu, T. Chen, A. Festag, H. Karl, G. Schäfer, and C. Fan. Secure, QoS-enabled mobility support for IP-based networks. In *Proc. IP Based Cellular Network Conference (IPCN)*, Paris, France, 2003.

[22] S. Ganeriwal, R. Kumar, and M. Srivastava. Timing-sync protocol for sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 138–149, 2003.

[23] A. Goldsmith. Stanford University EE 359 Wireless Communications Course Notes. http://www.stanford.edu/class/ee359/.

[24] P. Gupta and P. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory IT 2000*, IT-46(2):388–404, 2000.

[25] J. L. Hill and D. E. Culler. Mica: A wireless platform for deeply embedded networks. In *IEEE Micro*, pages 12–24, 2002.

[26] Y. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *8th ACM International Conference on Mobile Computing and Networking*, pages 12–23, September 2002.

[27] Y.C. Hu, A. Perrig, and D. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *Proceedings of IEEE Infocom 2003*, pages 1976–1986, 2003.

[28] Q. Huang, H. Kobayashi, and B. Liu. Modeling of distributed denial of service attacks in wireless networks. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, volume 1, pages 41–44, 2003.

[29] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networks (MobiCOM)*, August 2000.

[30] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-Efficient Comiputing for Wildlife Tracking: Design and Tradeoffs and Early Experiences with Zebranet. In *Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 96–107, 2002.

[31] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, 2003.

[32] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networks (MobiCOM)*, August 2000.

[33] S. Kay. *Fundamentals of Statistical Signal Processing: Detection Theory*. Prentice Hall, 1998.

[34] B. Kedem. *Time Series Analysis by Higher Order Crossings*. IEEE Press, 1994.

[35] L. Kleinrock. *Queueing Systems, Volume 2: Computer Applications*. John Wiley & Sons, 1976.

[36] L. Kleinrock and F. Tobagi. Packet switching in radio channels: Part i–carrier sense multiple-access modes and their throughput-delay characteristics. *IEEE Trans. on Communications*, 23(12):1400 – 1416, 1975.

[37] J. Kong, H. Luo, K. Xu, D. Gu, M. Gerla, and S. Lu. Adaptive security for multi-layer ad-hoc networks. *Special Issue of Wireless Communications and Mobile Computing*, 2002.

[38] P. Kyasanur and N.H. Vaidya. Detection and handling of mac layer misbehavior in wireless networks. In *Proceedings of the 2003 IEEE International Conference on Dependable Systems and Networks*, pages 173 – 182, 2003.

[39] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Comput. Networks*, 43(4):499–518, 2003.

[40] Y. Law, P. Hartel, J. den Hartog, and P. Havinga. Link-layer jamming attacks on s-mac. In *Proceedings of the 2nd European Workshop on Wireless Sensor Networks (EWSN 2005)*, pages 217 – 225, 2005.

[41] L. Lazos and R. Poovendran. SeRLoc: Secure range-independent localization for wireless sensor networks. In *Proceedings of the 2004 ACM Workshop on Wireless Security*, pages 21–30, 2004.

[42] M. Li, I. Koutsopoulos, and R. Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 1307–1315, 2007.

[43] Q. Li and W. Trappe. Relationship-based detection of spoofing-related anomalous traffic in ad hoc networks. In *Sensor and Ad Hoc Communications and Networks (SECON)*, pages 50–59, 2006.

[44] Z. Li, W. Trappe, Y. Zhang, and B. Nath. Securing wireless localization: Living with bad guys. In *DIMACS Workshop on Mobile and Wireless Security*, 2004.

[45] K. Ma, Y. Zhang, and W. Trappe. Mobile network management and robust spatial retreats via network dynamics. In *Proceedings of the The 1st International Workshop on Resource Provisioning and Management in Sensor Networks (RPMSN05)*, 2005.

[46] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *Proceedings of the Usenix Symposium on Operating Systems Design and Implementation*, 2002.

[47] M. Medard. Capacity of correlated jamming channels. In *Allerton Conference on Communications, Control and Computing*, pages 1043–1052, 1997.

[48] A. Nallanathan, Wang Feng, and H. K. Garg. Coexistence of wireless lans and bluetooth networks in mutual interference environment: An integrated analysis. *Comput. Commun.*, 30(1):192–201, 2006.

[49] V. Navda, A. Bohra, S. Ganguly, and D. Rubenstein. Using channel hopping to increase 802.11 resilience to jamming attacks. In *IEEE International Conference on Computer Communications(INFOCOM)*, pages 2526–2530, 2007.

[50] D. Nicelescu and B. Nath. DV based positioning in ad hoc networks. *Telecommunication Systems*, 22(1-4):267–280, 2003.

[51] D. L. Nicholson. *Spread Spectrum Signal Design LPD and AJ systems*. Computers Science Press, 1988.

[52] G. Noubir and G. Lin. Low-power DoS attacks in data wireless lans and countermeasures. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):29–30, 2003.

[53] S. Pack and Y. Choi. Pre-authenticated fast handoff in a public wireless lan based on ieee 802.1x model. In *Proceedings of the IFIP TC6/WG6.8 Working Conference on Personal Wireless Communications*, pages 175–182. Kluwer, B.V., 2002.

[54] P. Papadimittratos and Z. Haas. Secure routing for mobile ad hoc networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulations Conference (CNDS 2002), San Antonio*, 2002.

[55] D. Patrick and R. Morrow. *WiFi and Bluetooth Coexistence*. McGraw-Hill Professional, 2004.

[56] A. Perrig, R. Canetti, D. Song, and J.D. Tygar. Efficient and secure source authentication for multicast. In *Proceedings of Network and Distributed System Security Symposium*, February 2001.

[57] A. Perrig, R. Szewczyk, D. Tygar, V. Wen, and D. Culler. SPINS: security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002.

[58] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107. ACM Press, 2004.

[59] H. V. Poor. *An Introduction to Signal Detection and Estimation*. Springer Verlag, 2nd edition, 1994.

[60] B. Potter. Wireless security's future. *IEEE Security and Privacy Magazine*, 1(4):68–72, 2003.

[61] J. G. Proakis. *Digital Communications*. McGraw-Hill, 4th edition, 2000.

[62] A. Rajeswaran and R. Negi. Dos analysis of reservation based mac protocols. In *Proceedings of the IEEE International Conference on Communications*, 2005.

[63] A. Raniwala, K. Gopalan, and T. Chiueh. Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks. *ACM Mobile Computing and Communications Review*, 8(2):50–65, 2004.

[64] T.S. Rappaport. *Wireless Communications- Principles and Practice*. Prentice Hall, 2001.

[65] S. Ray, P. Moulin, and M. Medard. On jamming in the wideband regime. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2574–2577, 2006.

[66] M. Raya, J.P. Hubaux, and I. Aad. Domino: a system to detect greedy behavior in ieee 802.11 hotspots. In *MobiSYS '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 84–97. ACM Press, 2004.

[67] R.Garces and J. G. L. Aceves. Collision avoidance and resolution multiple access for multi-channel wireless networks. In *Proceedings of IEEE INFOCOM*, pages 595–602, 2000.

[68] J. A. Salehi. Code division multiple-access techniques in optical fiber networks.i. fundamental principles. *IEEE Transactions on Communications*, 37:824–833, 1989.

[69] C. Schleher. *Electronic Warfare in the Information Age*. MArtech House, 1999.

[70] M. K. Simon, J. K. Omura, R. A. Scholts, and B. K. Levitt. *Spread Spectrum Communications Handbook*. McGraw-Hill Inc, revised edition edition, 1994.

[71] B. Sklar. *Digital Communications: Fundamentals and Applications*. Prentice Hall, 2nd edition, 2001.

[72] J. So and N. Vaidya. Multi-channel MAC for ad hoc network: Handling multi-channel hidden terminals using a single transceiver. In *Proceedings of ACM MobiHoc*, pages 222 – 233, 2003.

[73] W. Trappe and L. Washington. *Introduction to Cryptography with Coding Theory*. Prentice Hall, 2002.

[74] A. Weimerskirch and G. Thonet. A distributed light-weight authentication model for ad-hoc networks. In *The 4th International Conference on Information Securtiy and Cryptology (ICISC 2001)*, December 2001.

[75] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27, 2003.

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

[77] A. Wood, J. Stankovic, and S. Son. JAM: A jammed-area mapping service for sensor networks. In *24th IEEE Real-Time Systems Symposium*, pages 286 – 297, 2003.

[78] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 46–57, 2005.

[79] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel surfing and spatial retreats: defenses against wireless denial of service. In *Proceedings of the 2004 ACM workshop on Wireless security*, pages 80 – 89, 2004.

[80] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the IEEE INFOCOM*, volume 3, pages 1567– 1576, 2002.

[81] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 1–13, 2003.

[82] L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.

[83] S. Zhu, S. Xu, S. Setia, and S. Jajodia. LHAP: A lightweigth hop-by-hop authentication protocol for ad-hoc networks. In *International Workshop on Mobile and Wireless Network (MWN 2003)*, Mai 2003.

Curriculum Vitae

Wenyuan Xu

1998 BS in Computer Science; Zhejiang University, Hangzhou, China.

2001 MS in Computer Engineering; Zhejiang University, Hangzhou, China.

2007 Ph.D. in Computer Engineering; Rutgers University, NJ, USA.

1998-2001 Research Assistant, Computer Vision Laboratory, Zhejiang University, Hangzhou, China

2002 Research Assistant, Department of Computer Science, Rensselaer Polytechnic Institute, NY, USA

2005 Research Intern, Hewlett-Packard Labs, NJ, USA

2003-2007 Graduate Assistant, WINLAB, Rutgers University, NJ, USA.

Publications

2004 *Channel surfing and spatial retreats: defenses against wireless denial of service*. W. Xu, T. Wood, W. Trappe, and Y. Zhang, in the Proceedings of the 2004 ACM Workshop on Wireless security (WiSe), pg. 80-89, 2004.

Key Management for 3G MBMS Security. W. Xu, W. Trappe, and S. Paul, in the Proceedings of IEEE Global Telecommunications Conference (GLOBECOM), Vol. 4, pg. 2276-2280, 2004.

2005 *The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks*. W. Xu, W. Trappe, Y. Zhang and T. Wood, in the Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc), pg. 46-57, 2005.

2006 *Poster Abstract: Channel Surfing: Defending Wireless Sensor Networks from Jamming and Interference*. W. Xu, W. Trappe, and Y. Zhang, in the Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (SenSys), pg. 403-404, 2006.

Securing Wireless Systems via Lower Layer Enforcements. Z. Li, W. Xu, R. Miller and W. Trappe, in the Proceedings of the 2006 ACM Workshop on Wireless Security (WiSe), pg. 33-42, 2006.

TRIESTE: A Trusted Radio Infrastructure for Enforcing SpecTrum Etiquettes. W. Xu, P. Kamat and W. Trappe, in the Proceedings of the IEEE Workshop on Networking Technologies for Software Defined Radio (SDR) Networks (Held in Conjunction with IEEE SECON), 2006.

Jamming Sensor Networks: Attack and Defense Strategies. W. Xu, K. Ma, W. Trappe, and Y. Zhang, IEEE Networks Special Issue on Sensor Networks, Vol. 20, No. 3, pg. 41-47, May/June 2006.

2007 *Service Discovery and Device Identification in Cognitive Radio Networks*. R. Miller, W. Xu, P. Kamat, and W. Trappe, in the Proceedings of the 2nd IEEE Workshop on Networking Technologies for Software Defined Radio (SDR) Networks (Held in Conjunction with IEEE SECON), 2007.

Defending Wireless Sensor Networks from Radio Interference through Channel Adaptation. W. Xu, W. Trappe, and Y. Zhang, submitted to ACM Transactions on Sensor Networks (TOSN)

Temporal Privacy in wireless sensor networks. P. Kamat, W. Xu, Y. Zhang and W. Trappe, in the Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS2007), 2007.

Channel Surfing: Defending Wireless Sensor Networks from Jamming and Interference. W. Xu, W. Trappe, and Y. Zhang, in the Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN07), pg. 499-508, 2007.